Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Uporabniška programska oprema

**Matej Artač, Borut Batagelj, Matjaž Jogan,
Žiga Kranjec, Bojan Kverh, Katarina Mele,
Peter Peer, Miha Peternel, Franc Solina**

Ljubljana, 2004

CIP – Kataložni zapis o publikaciji
Narodna in univerzitetna knjižnica, Ljubljana

004.4(075.8)

UPORABNIŠKA programska oprema / Matej Artač ... [et al.]. - 3. dopolnjena izd. - Ljubljana : Fakulteta za računalništvo in informatiko, 2004

ISBN 961-6209-48-5
1. Artač, Matej
216137728

Recenziral: doc. dr. Aleš Jaklič

Založba: Fakulteta za računalništvo in informatiko, 2004
Urednik: mag. Peter Šega

Natisnil: FORMATISK Ljubljana
Naklada: 800 izvodov
3. dopolnjena izdaja

Kazalo

Slike

Tabele

Predgovor

Obvladovanje računalnikov, predvsem pisanje razvnovrstnih besedil, priprava grafičnega gradiva, komuniciranje z elektronsko pošto in iskanje informacij na svetovnem spletu, se danes zahteva od vedno večjega števila ljudi na vseh možnih področjih dela. Računalniška pismenost pa je tudi podlaga uspešnemu študiju na vseh zahtevnostnih ravneh in na vseh študijskih področjih.

Knjiga "Uporabniška programska oprema" obravnava tista osnovna računalniška znanja, ki v sodobnem svetu sodijo skoraj že v sklop splošne pismenosti. Učbenik je v prvi vrsti namenjen študentom prvega letnika študija računalništva in informatike, da bi osvežili in poenotili svoje znanje uporabe osnovnih programskih orodij, vendar pa tudi vsem drugim, ki se želijo naučiti učinkovitega dela z računalnikom. Posebnega predznanja računalništva knjiga ne zahteva.

Knjiga je sestavljena iz enajstih poglavij, ki obravnavajo posamezne sklope programskih orodij. Prvi del vsakega poglavja bralca uvede v splošno problematiko določenega področja uporabe, drugi del poglavja pa obravnava eno ali dve značilni programski orodji za obravnavano uporabniško področje. Bralcu knjiga za vsako izbrano programsko orodje ponuja dovolj osnovnih informacij, da lahko začne samostojno uporabljati to orodje. Tisti bralci, ki bi radi bolj podrobno spoznali določeno uporabniško področje, pa lahko posežejo po drugih bolj specializiranih knjigah.

Pri izbiri programskih orodij smo dali prednost brezplačnim in javno dostopnim programom, ki so za veliko večino uporabnikov povsem enakovredni mnogo dražjim plačljivim programom. Prehod na podobne,

komercialno zasnovane programe pa je zelo enostaven.

V prvem uvodnem poglavju knjiga bralce na kratko seznani z zgodovino računalnikov, njihovo zgradbo in s predstavitvijo informacij v računalniku.

Drugo poglavje seznani bralca s pravno zaščito programske opreme, saj ravno pri uporabi najbolj razširjenih programskih orodij prihaja do pogostih kršitev avtorskih pravic.

Tretje poglavje govori o različnih vrstah komuniciranja uporabnika z računalnikom, od ukaznega načina komuniciranja do uporabe grafičnih uporabniških vmesnikov. Kot primera grafičnega uporabniškega vmesnika to poglavje predstavi *Slix* in pisarniško okolje *OpenOffice.org*

Četrto poglavje je namenjeno operacijskim sistemom. Kot primer operacijskega sistema obravnavamo *Linux*, in sicer z ukazno vrstičnim ter grafičnim uporabniškim vmesnikom. Posebej obravnavamo *Slix* kot vrsto Živega Linuxa, ki se nahaja tudi na zgoščenki, ki je priložena tej knjigi.

Peto poglavje je namenjeno urejanju besedil. V uvodu je nekaj osnovnih napotkov o oblikovanju besedil in izbiri pisav ter pisanju strokovnih besedil. Kot primer vizualnega urejevalnika smo izbrali *Writer* iz okolja *OpenOffice.org*, kot primer logičnega urejanja besedila pa *LaTeX*.

Šesto poglavje je namenjeno preglednicam in orodju *Calc* iz okolja *OpenOffice.org*.

Sedmo poglavje obravnava računalniško grafiko. Kot predstavnik rastrskega grafičnega orodja je predstavljeno orodje *GIMP*, kot primer vektorskega grafičnega orodja pa Draw iz okolja *OpenOffice.org*. Na kratko je predstavljen še paket za risanje diagramov *GraphViz*.

V osmem poglavju govorimo o tem, kako pripravimo govorno predstavitev. Kot primer orodja za izdelavo prosojnic smo zopet izbrali orodje iz okolja *OpenOffice.org* in sicer orodje *Impress*.

Deveto poglavje obravnava matematične programe. Predstavljeni sta orodji *Mathematica*, ki je v prvi vrsti namenjena simboličnemu reševanju matematičnih problemov, in *Matlab*, ki je namenjen numeričnemu reševanju matematičnih nalog.

Deseto poglavje je v celoti namenjeno medmrežju. V uvodnem delu so podani osnovni napotki za vsebinsko in oblikovno izdelavo spletnih predstavitev. Opisano je delo s spletnim brskalnikom in odjemalcem elektronske pošte. V zadnjem delu poglavja pa obravnavamo jezika HTML in XML za izdelavo spletnih strani.

Enajsto poglavje na osnovi praktičnega primera pokaže, kako lahko različna orodja učinkovito kombiniramo tako, da združujemo oziroma nadgrajujemo njihove rezultate.

V dodatku so napotki za delo s Slixom in namestitev operacijskega sistema Linux ter kratek angleško-slovenski slovarček najpogostejših

računalniških izrazov s področja uporabniške programske opreme.

Bralec, ki ga zanima le določeno uporabniško področje, lahko prebere le odgovarjajoča poglavja. Tako lahko na primer nekdo, ki ga zanimajo le pisarniški programi, prebere le poglavja 5, 6, 7 in 8.

Na koncu bi se rad zahvalil za trud pri nastanku te knjige tudi mojim soavtorjem, ki v veliki večini tudi vodijo vaje pri mojih predmetih. Hvala tudi Ljudmili in prav posebej Žigi Kranjcu za pomoč v zvezi s Slixom.

V Ljubljani, oktober 2004 FRANC SOLINA

1

Uvod

Obvladanje dela z računalnikom se danes pričakuje od vedno večjega števila zaposlenih. Obvladanje osnovnih računalniških spretnosti, kot so pisanje besedil, komuniciranje in iskanje informacij preko interneta, pa postaja pomembnejše tudi za splošno pismenost in informiranost ljudi. Nenazadnje so te spretnosti osnova uspešnemu študiju na vseh zahtevnostnih ravneh in na vseh študijskih področjih [1].

S pojmom računalnik označujemo stroj, na katerem se izvršujejo programi, ki so abstrakten zapis postopkov ali algoritmov v strojnem jeziku. Računalnike lahko delimo po več kriterijih, na primer po namenu (specializirani, večnamenski), tehnologiji izdelave (mehanski principi, elektronke, tranzistorji, integrirana vezja), po principu delovanja (ukazno krmiljeni, podatkovno krmiljeni, skladovni, asociativni), ipd.

Specializirani računalniki se ponavadi uporabljajo za reševanje točno določenega problema ali tipa problemov. Običajno jih ne moremo uporabljati za reševanje problemov, za katere niso predvideni. Take računalnike najdemo v raznih napravah, kot so avtomobili, razni gospodinjski aparati, videorekorderji, TV sprejemniki, kamere, mobilni telefoni ipd.

Večnamenskim računalnikom lahko spreminjamo program, ki ga izvajajo, in jih tako lahko uporabimo za reševanje splošnejših problemov. Specializirani računalniki so ponavadi cenejši in učinkovitejši za reševanje problema, za katerega so programirani, kot pa večnamenski.

V tem poglavju je kratek oris zgodovine razvoja računalnikov, opisana je osnovna zgradba računalnikov in opisan je način, kako so v računalnikih predstavljene informacije.

1.1 Kratka zgodovina računalnikov

Prvi poskusi izdelave strojev, ki bi omogočali avtomatizacijo rutinskih opravil, segajo v 17. stoletje. V prvo generacijo sodijo mehanski stroji, ki so delovali na principu zobatih koles in zobatih trakov. Take stroje so skonstruirali npr. Wilhelm Schichard, Blaise Pascal in Gottfried Wilhelm Leibniz. Ti stroji so lahko opravljali preproste aritmetične operacije, kot so seštevanje, odštevanje, množenje in deljenje. Take stroje srečamo še danes v avtomobilskih in električnih števcih. V to kategorijo sodi tudi stroj Charlesa Babbagea, ki je bil namenjen izračunu logaritemskih tablic. Charles Babbage je tudi postavil temelje arhitekture modernih računalnikov.

Prvi programirni mehanski stroj imenovan Harvard Mark I so izdelali na harvardski univerzi, uporabljali pa so ga v mornarici. Programirati se ga je dalo z luknjanim trakom.

Naslednja generacija so elektronski računalniki, zgrajeni s pomočjo elektronk, relejev in stikal. Leta 1943 so na univerzi v Manchestru izdelali specializirani elektronski računalnik Colossus, ki je bil namenjen dešifriranju kodiranih sporočil. Prvi večnamenski elektronski računalnik je bil ENIAC, ki so ga v letih 1943–1946 pod vodstvom Presperja Eckerta in Johna Mauchlyja razvili na University of Pennsylvania v Philadelphiji [3]. Vseboval je okrog 18000 elektronk in 6000 relejev, njegova prostornina pa je bila 90 kubičnih metrov. Programirati se ga je dalo s pomočjo stikal in povezovanja električnih kablov.

Naslednji korak v razvoju so bili računalniki, ki so imeli program in podatke shranjene v skupnem pomnilniku. Tako zasnovo danes imenujemo von Neumannova arhitektura, po Johnu von Neumannu, avtorju poročila iz leta 1945, v katerem je opisal zasnovo takšnega računalnika. Vendar se je skupnega pomnilnika prvi domislil Presper Eckert, ko je načrtoval ENIAC-ovega naslednika, kjer je predvidel shranjevanja ukazov v pomnilnik, da bi se dalo računalnik lažje in hitreje programirati [3]. Prvi računalniki s tako zasnovo so bili Manchester Mark I, izdelan leta 1948 na univerzi v Manchestru, EDSAC, izdelan leta 1949 na univerzi v Cambridgu, ter EDVAC, ki so ga izdelali leta 1952 na univerzi v Princetonu, predvsem pa prvi komercialni računalnik UNIVAC, ki je prišel na tržišče leta 1951. UNIVAC je izdelalo podjetje, ki sta ga ustanovila Eckert in Mauchly in ki ga je kasneje prevzelo podjetje Sperry Rand.

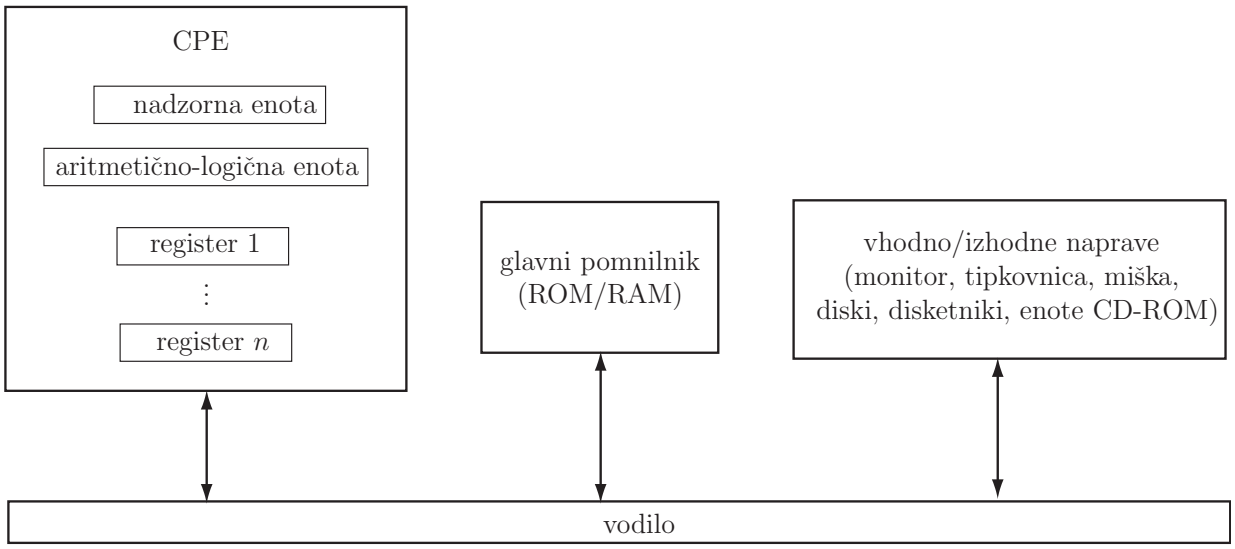
V tretjo generacijo prištevamo računalnike, zgrajene z integriranimi vezji. Nadomestitev tranzistorjev z integriranimi polprevodniškimi vezji je omogočila večjo hitrost in manjšo porabo energije. Pojavile so se mikroprogramirane centralno procesne enote in integrirana polprevodniška pomnilniška vezja. Tipični primeri takih računalnikov so bili CDC 6000, DEC PDP-11, Honeywell 6000 ter IBM serije 360 in 370.

Pri teh računalnikih se je prvič pojavil operacijski sistem. To je poseben program ali skupina programov, ki skrbi za nadzor delovanja ostalih programov, komunikacijo z vhodno/izhodnimi napravami in izrabo virov. Prav tako se je pri tej generaciji računalnikov prvič pojavilo izvajanje več programov hkrati s pomočjo razdeljevanja procesorskega časa med programe. Prvi računalniki tretje generacije so nastali okoli leta 1965.

Obdobje četrte generacije računalnikov se začenja z nastankom prvega mikroprocesorja Intel 4004 leta 1971 in traja še danes. Mikroprocesorji so centralno procesne enote, ki so realizirani na enem samem čipu. To omogoča visoka stopnja integracije elektronskih komponent. Po mikroprocesorju Intel 4004, ki je bil 4-bitni procesor, so se pojavili še 8-bitni Intel 8008, Intel 8080 in Motorola 6800. Število izdelovalcev integriranih vezij je raslo in temu primerno je padala njihova cena. Zato so se v 80ih letih pojavili ceneni osebni računalniki (Sinclair ZX80 in ZX81, Commodore 16 in 64, CPC Amstrad ipd.), ki so omogočili nakup računalnikov tudi širšim množicam. Žal pa so za resno delo imeli premajhen pomnilnik, ki ga ni bilo mogoče razširiti, zato so se večinoma uporabljali za igranje računalniških igric. Kasneje so se pojavili računalniki IBM PC XT in AT, ki so postavili temelje, na katerih sloni večina računalnikov še danes. Predvsem je tu pomemben razvoj standardov za različne dele računalnikov (grafične kartice, mrežne kartice, pomnilnik ipd.), ki omogočajo, da posamezno komponento računalnika nadomestimo z drugo, boljšo, ne da bi bilo treba zamenjati kak drug del računalnika ali celo cel računalnik.

V podjetju Xerox so že sredi 70ih let v računalnika Alto in Star prvič vgradili operacijski sistem z grafičnim uporabniškim vmesnikom. Prvega niso nikoli dali v prodajo, drugi pa je na trg prišel leta 1981, vendar je bil predrag, da bi bil lahko uspešen. Njihove ideje so uporabili pri podjetju Apple in leta 1984 izdelali računalnik Macintosh, ki je prvi komercialno uspešen računalnik z grafičnim uporabniškim vmesnikom. Grafični uporabniški vmesniki so predvsem po zaslugi Macintosha postali standard. Podjetje Microsoft je izdalo več vrst operacijskih sistemov Windows za osebne računalnike PC, ki imajo vsi vgrajen grafični uporabniški vmesnik. Prav tako pa je ta vgrajen tudi v večino operacijskih sistemov UNIX na delovnih postajah in Linux na osebnih računalnikih (X-Window). Pomemben je tudi razvoj večopravilnih in večuporabniških računalnikov, na katerih lahko delo preko terminalov hkrati opravlja več uporabnikov.

Peta generacija računalnikov naj bi omogočala logično programiranje v stilu jezikov LISP in Prolog, vendar pa taki načini zaenkrat še niso najbolj uspešni in razširjeni.



Slika 1.1: Sestavni deli računalnika

1.2 Zgradba računalnikov

Dandanašnji računalniki so običajno sestavljeni iz centralno procesne enote (CPE), pomnilnika, vhodno/izhodnih (V/I) naprav ter vodil, ki služijo za prenos podatkov med deli računalnika (slika 1.1).

CPE je glavni del vsakega računalnika. Njena naloga je, da bere in izvršuje ukaze. CPE je običajno sestavljena iz večjega števila enot in registrov. Vsaka enota ima točno določeno funkcijo; v CPE so ponavadi posebne enote za branje ukazov in operandov iz pomnilnika, enote za izvrševanje aritmetično logičnih operacij in registri za shranjevanje ukazov in operandov.

Delovanje CPE lahko opišemo kot zaporedje ukaznih ciklov. En ukazni cikel je sestavljen iz branja ukaza iz pomnilnika, dekodiranja kode ukaza, morebitnega branja operandov iz pomnilnika (če ukaz to zahteva), posredovanja ukaza in operandov ustrezni enoti in izvrševanja ukaza v tej enoti. V novejših CPE se lahko naenkrat izvaja tudi več ukaznih ciklov (cevovodno procesiranje). Nabori ukazov, ki jih posamezna CPE razume, so zelo raznoliki tako po samih ukazih kot tudi po številu le-teh.

Nekateri ukazi kot operande zahtevajo podatke iz registrov namesto iz pomnilnika. CPE hrani v registrih različne informacije. Prednost uporabe registrov je v hitrosti, saj se podatki iz registrov hitreje prenesejo v posamezne enote CPE kot pa podatki iz pomnilnika, kjer je potreben prenos preko vodil. Število registrov v CPE je omejeno s ceno, saj morajo biti realizirani s kar se da hitrimi vezji, ki pa so zelo draga. Vsaka CPE vsebuje programski števec, v katerem je naslov v pomnilniku, iz katerega

se bo prebral naslednji ukaz. Prav tako se v CPE nahaja eden ali več registrov stanja, ki odražajo posebne rezultate po zadnjem izvršenem ukazu. Vsaka CPE ima enega ali več splošno namenskih registrov, ki hranijo podatke za ukaze in rezultate določenih ukazov. Obstajajo tudi posebni registri, ki jih lahko uporabljajo samo določeni ukazi, na primer registri za posredno in indeksno naslavljanje pomnilniških lokacij. Predvsem novejše CPE vsebujejo tudi registre za računanje z racionalnimi števili in števili s plavajočo vejico.

Pomnilnik sestavlja več pomnilniških lokacij, ki hranijo podatke v obliki pomnilniških besed. Poznamo pomnilnike, iz katerega je mogoče samo brati (ROM - read only memory), in pomnilnike, ki omogoča tako branje kot zapisovanje (RAM - random access memory). Vsaka pomnilniška lokacija ima svoj naslov in vsebuje eno samo pomnilniško besedo ali podatek. Do podatka lahko pridemo samo tako, da pomnilniku preko vodila posredujemo naslov pomnilniške lokacije, pomnilnik pa nam zopet preko vodila vrne podatek, ki se nahaja na tej lokaciji. Na podoben način lahko podatke v pomnilnik tudi zapisujemo, le da tokrat poleg naslova pomnilniku posredujemo še podatek. Obstajajo tudi asociativni pomnilniki, ki so naslovljivi z vsebino, vendar se ne uporabljajo v večnamenskih računalnikih.

Velikost pomnilnika merimo v pomnilniških besedah, ki vsebujejo enega ali več bitov, bit pa je osnovna enota, ki ima lahko samo dve logični vrednosti (0 ali 1). V praktično vseh današnjih računalnikih pomnilniška beseda vsebuje 8 bitov, velikosti pomnilnikov pa so v obsegu nekaj milijonov pomnilniških besed.

Vhodno/izhodne naprave so nujno potrebne za udobno delo z računalnikom. Vhodne enote so tiste, s katerimi podatke posredujemo računalniku, medtem ko so izhodne enote tiste, katerim računalnik posreduje podatke. Med tipične vhodne naprave sodijo tipkovnica, miška, svetlobno pero, igralna palica, optični bralnik, bralnik enot CD-ROM ipd. Med tipične izhodne naprave štejemo monitor, tiskalnik, zvočnike, risalnik ipd. Vgrajeni in prenosni mediji (disketniki, trdi diski, tračne enote) imajo značilnosti tako vhodnih kot izhodnih naprav.

Vodilo je namenjeno prenosu podatkov med pomnilnikom, CPE in V/I napravami. Ponavadi je sestavljeno iz naslovnega, podatkovnega in nadzornega dela. Na naslovni del vodila CPE ali V/I naprava vpiše naslov želene pomnilniške lokacije. Po podatkovnem delu se prenašajo podatki iz pomnilnika oziroma v pomnilnik. Z nadzornim delom lahko pomnilnik signalizira, kdaj je na podatkovnem delu res želeni podatek (branje iz pomnilnika) oziroma kdaj je posredovani podatek zapisan v pomnilnik (pisanje v pomnilnik), zagotavlja, da ni prišlo do napake pri prenosu preko vodila, ipd. Širina podatkovnega dela vodila v bitih je ponavadi večkratnik dolžine pomnilniške besede v bitih, širina naslovnega dela pa

taka, da je možno nedvoumno nasloviti vse pomnilniške lokacije.

1.3 Predstavitev podatkov v računalniku

V samem začetku razvoja računalnikov so se pojavljali računalniki, v katerih so bila števila predstavljena v desetiškem številskem sestavu (npr. ENIAC), vendar pa je sčasoma, predvsem zaradi tehnologije pomnilniških vezij, prevladala binarna oblika. V taki obliki so predstavljeni ukazi, števila in neštevilčni podatki. Vsak podatek, ki je shranjen na neki pomnilniški lokaciji, je sestavljen iz toliko bitov, kot je dolžina pomnilniške besede. Iz podatka samega ni razvidno, kaj predstavlja. Če CPE prebere podatek iz pomnilniške lokacije, kjer bi se moral nahajati ukaz (glede na vrednost registra, kjer se hrani naslov naslednjega ukaza), potem je zanjo to pač ukaz, v nasprotnem primeru pa je to operand. Interpretacija pomena operandov je prepuščena uporabniškim programom. V računalniku lahko predstavimo tako cela kot tudi realna števila. Cela števila lahko predstavimo kot nepredznačena ali predznačena, za realna števila pa uporabljamo zapis v fiksni ali plavajoči vejici. Neštevilčni podatki so lahko besedilo, slike in zvok, ki so tudi številčni podatki, in so običajno precej daljši od ene pomnilniške besede. Zato je v eni pomnilniški besedi shranjen le majhen del takega podatka.

1.3.1 Predstavitev numeričnih podatkov

Predstavitev nepredznačenih celih števil

Nepredznačena cela števila so v računalniku predstavljena z dvojiškim zapisom njihove vrednosti. Če je pomnilniška beseda dolga 8 bitov, potem lahko na eni pomnilniški lokaciji predstavimo nepredznačena cela števila med vključno 0 (dvojiško 00000000) in 255 (dvojiško 11111111). Število 76 je na primer predstavljeno kot 01001100. Če želimo predstaviti več nepredznačenih števil, moramo uporabiti več pomnilniških lokacij. S 16 biti namreč lahko predstavimo cela števila od 0 do 65535, z 32 biti pa cela števila od 0 do 4294967295. V splošnem torej lahko z n biti predstavimo nepredznačena cela števila od 0 do $2^n - 1$.

Predstavitev predznačenih števil

Predznačena števila so lahko negativna ali nenegativna. Slednja so predstavljena tako kot nepredznačena, le da mora biti prvi bit, ki je namenjen predznaku števila, postavljen na nič. Za predstavitev negativnih števil pa se uporabljata eniški ali dvojiški komplement. Eniški komplement števila dobimo tako, da zapišemo njegovo absolutno vrednost

v dvojiški številski sestav, tu pa enice zamenjamo z ničlami in obratno. Torej, če bi hoteli zapisati -84 v eniškem komplementu z 8 biti, potem bi najprej zapisali 84 v dvojiškem številskem sestavu, kar pomeni 01010100, nato pa bi zamenjali enice z ničlami in obratno, tako da bi dobili 10101011. Vendar eniški komplement ni najbolj primeren za predstavitev predznačenih števil, saj imamo število 0 predstavljeno z dvema kodama in sicer $00\ldots0$ ter $11\ldots1$ (to je koda števila -0, ki je enako 0). Taka predstavitev je neustrezna (s stališča prevajalnikov višjih programskih jezikov, saj bi to lahko malo povečalo in upočasnilo programsko opremo), zato se v praksi uporablja dvojiški komplement, ki je eniški komplement, povečan za 1, pri čemer se bit na mestu $n+1$ (prelivni bit), če je n število bitov za predstavitev predznačenih števil, zanemari. Število -84 bi v dvojiškem komplementu z 8 biti izgledalo torej kot 10101100. Število -1 bi tako predstavili z 11111111, -128 pa kot 10000000.

Negativna števila, predstavljena v dvojiškem komplementu, imajo vedno prvi (najpomembnejši) bit enak 1, nenegativna pa 0. V splošnem lahko v dvojiškem komplementu z n biti predstavimo cela števila od vključno -2^{n-1} (predstavljeno kot $100\ldots0$) do $2^{n-1}-1$ (predstavljeno kot $011\ldots1$).

Potrebno je upoštevati, da če računalnik prebere iz pomnilnika število 10101001, ne more vedeti, ali gre za predznačeno število (-87) ali za nepredznačeno (169) ali pa za nekaj čisto drugega (del besedila, slike, zvoka ipd.). To, da bo program pravilno interpretiral podatek, ki ga je prebral iz pomnilnika, mora biti poskrbljeno v programu samem.

Predstavitev realnih števil v računalniku

Realna števila so sestavljena iz celega in decimalnega dela. Glavni problem pri predstavitvi realnih števil je ta, da imajo ta lahko neskončno število decimalk, računalniki pa imajo seveda samo končen pomnilnik. Zato so realna števila predstavljena s svojimi približki.

Zapis s fiksno vejico namenja za predstavitev celega in decimalnega dela števila fiksno število bitov, pri čemer število bitov, namenjenih za celi del, ni nujno enako številu bitov, namenjenih za decimalni del. Celi del števila je predstavljen tako, da je absolutna vrednost celega dela števila zapisana kot nepredznačena cela števila. Neceli del števila se pretvori v zapis s fiksno vejico tako, da se ga v vsakem koraku množi z dva, zapiše celi del, ta celi del se od števila odšteje in to se ponavlja dokler ni zapisanih toliko bitov, kolikor jih je namenjeno za decimalni del. Spodnji primer ilustrira, kako zapišemo neceli del števila 17.64 z 8 biti:

$$0.64 \cdot 2 \;=\; 1.28 \qquad \text{zapišemo 1 in jo odštejemo od števila 1.28} \qquad (1.1)$$

$0.28 \cdot 2 = 0.56$ zapišemo 0
$0.56 \cdot 2 = 1.12$ zapišemo 1 in jo odštejemo od števila 1.12
$0.12 \cdot 2 = 0.24$ zapišemo 0
$0.24 \cdot 2 = 0.48$ zapišemo 0
$0.48 \cdot 2 = 0.96$ zapišemo 0
$0.96 \cdot 2 = 1.92$ zapišemo 1 in jo odštejemo od števila 1.92
$0.92 \cdot 2 = 1.84$ zapišemo 1 in končamo, ker imamo že 8 bitov

Neceli del števila 17.64 se z 8 biti torej predstavi kot 10100011 (strogo matematično bi lahko zapisali $0.64_{(10)} \approx 0.10100011_{(2)}$). Če je za celi del namenjenih 8 bitov, potem se celi del predstavi kot 00010001. Število 17.64 predstavimo v zapisu s fiksno vejico, kjer je 8 bitov namenjenih za celi del in 8 bitov za decimalni del, kot 00010001.10100011.

Negativno število je v zapisu s fiksno vejico predstavljeno z dvojiškim komplementom tistega kar bi bila predstavitev njegove absolutne vrednosti v istem zapisu. Na primer, število -18.75 ima absolutno vrednost 18.75, to pa bi bilo v zapisu s fiksno vejico, kjer je 8 bitov za celi del in 8 bitov za decimalni del, predstavljeno kot 00010010.11000000. Dvojiški komplement tega je 11101101.01000000, to pa je tudi predstavitev števila -18.75 v istem zapisu. Če imamo n bitov za celi del in m bitov za decimalni del, lahko v zapisu s fiksno vejico zapišemo števila med -2^{n-1} in $2^{n-1} - 2^{-m}$. Najmanjše pozitivno število, ki ga lahko v takem zapisu predstavimo točno, pa je 2^{-m}.

Zapis s plavajočo vejico se danes uporablja precej bolj pogosto kot zapis s fiksno vejico, ker je bolj prilagodljiv. Zapis s fiksno vejico namreč ne upošteva dejstva, da so pri velikih številih decimalna mesta ponavadi manj pomembna, pri zelo majhnih (pozitivnih) številih pa je večina bitov pri predstavitvi s fiksno vejico enaka 0. Pri zapisu s plavajočo vejico je ponavadi 1 bit namenjen predznaku števila, e bitov je namenjenih za eksponent, ter m bitov za mantiso. Vsako realno število lahko zapišemo v obliki

$$(-1)^p \cdot 1.w \cdot 2^z. \tag{1.2}$$

Če ima p lahko samo vrednosti 0 (nenegativno število) in 1 (negativno število), potem ga lahko interpretiramo kot predznak števila. Celo število z ima vlogo eksponenta, katerega vrednost, povečana za $2^{e-1} - 1$ je v zapisu s plavajočo vejico zapisana kot nepredznačeno celo število. Število w ima vlogo mantise (necelega dela števila). Mantiso zapišemo v zapisu s plavajočo vejico tako kot pri zapisu s fiksno vejico. Razčlenitev zapisa v plavajoči vejici na bite, kjer je en bit namenjen predznaku, e bitov za eksponent in m bitov za mantiso, je prikazana na sliki 1.2.

p	y_1	...	y_e	x_1	...	x_m

Slika 1.2: Zapis s plavajočo vejico, kjer p predstavlja predznak števila, y je dvojiški zapis števila $z + 2^{e-1} - 1$, x pa je dvojiški zapis decimalk w tako kot pri zapisu s fiksno vejico.

Kot primer lahko navedemo pretvorbo števila 7.5 v zapis s plavajočo vejico, kjer je 1 bit namenjen predznaku, 7 bitov za eksponent in 8 za mantiso. V skladu z enačbo (1.2) se da število 7.5 zapisati kot $(-1)^0 \cdot 1.875 \cdot 2^2$. Predznak je tako predstavljen z 0, za eksponent moramo v dvojiškem številskem sestavu zapisati število $2 + 2^{7-1} - 1$ (ker je 7 bitov namenjenih za eksponent), za mantiso pa moramo 0.875 zapisati tako kot smo to storili v enačbi (1.1). Tako je število 7.5 v zapisu s plavajočo vejico, kjer je 1 bit namenjen za predznak, 7 za eksponent in 8 za mantiso zapis 0 1000001 11100000 (presledki ločijo bite, namenjene za predznak, eksponent in mantiso).

Število -0.08 lahko zapišemo kot $(-1)^1 \cdot 1.28 \cdot 2^{-4}$. V zapisu s plavajočo vejico, kjer je 1 bit namenjen za predznak, 8 za eksponent in 15 za mantiso, je to število tako zapisano kot 1 01111011 010001111010111. Število je negativno, zato je predznak predstavljen z 1, za eksponent smo zapisali število $-4 + 2^{8-1} - 1$ v dvojiškem številskem sestavu, za mantiso pa smo število 0.28 zapisali po postopku opisanem v enačbi (1.1).

V današnjih računalnikih se za predstavitev realnih števil najpogosteje uporabljajo zapisi s plavajočo vejico, ki jih definira standard IEEE 754. Njihovo razčlenitev na bite podaja tabela 1.1.

zapis	*enojna natančnost*	*dvojna nat.*	*razširjena nat.*
št. bitov za predznak	1	1	1
št. bitov za eksponent	8	11	15
št. bitov za mantiso	23	52	64
skupaj bitov	32	64	80

Tabela 1.1: Razčlenitev zapisov enojne, dvojne in razširjene natančnosti standarda IEEE 754

V splošnem lahko s zapisom v plavajoči vejici, kjer je 1 bit namenjen za predznak, e bitov za eksponent in m za mantiso, predstavimo števila med $-(2 - 2^{-m}) \cdot 2^{2^{e-1}}$ do $(2 - 2^{-m}) \cdot 2^{2^{e-1}}$. Najmanjše pozitivno število, ki ga lahko v takem zapisu predstavimo točno, pa je $2^{-2^{e-1}+1}$.

1.3.2 Predstavitev neštevilčnih podatkov

Predstavitev besedila

Besedilo je v računalniku predstavljen tako, da je vsak posamezen znak, ki ga sestavlja, predstavljen z določeno kodo. Kadar se besedilo nahaja v pomnilniku, se kode sosednih znakov v besedilu ponavadi nahajajo tudi v sosednih pomnilniških lokacijah. Kodo posameznega znaka določa kodna tabela. Te so zelo različne, prevladuje pa 7-bitna kodna tabela po standardu ASCII. Kode nekaterih črk, številk in posebnih znakov, upoštevajoč ta standard, so podane v tabeli 1.2.

znak	*koda*	*znak*	*koda*	*znak*	*koda*	*znak*	*koda*
!	33	8	56	G	71	S	83
(	40	9	57	H	72	T	84
)	41	:	58	I	73	U	85
+	43	=	61	J	74	V	86
0	48	?	63	K	75	W	87
1	49	@	64	L	76	X	88
2	50	A	65	M	77	Y	89
3	51	B	66	N	78	Z	90
4	52	C	67	O	79	^	94
5	53	D	68	P	80	a	97
6	54	E	69	Q	81	b	98
7	55	F	70	R	82	~	126

Tabela 1.2: Izvleček iz kodne tabele po standardu ASCII

7-bitna kodna tabela po standardu ASCII vsako kodo zapiše kot nepredznačeno celo število s 7 biti. V pomnilniku, kjer je pomnilniška beseda dolga 8 bitov, se kode ASCII zapišejo na najmanj pomembnih 7 bitov, najpomembnejši bit pa je postavljen na 0, tako da se dvojiška vrednost kode ohrani. V takem primeru bi bila beseda “UPO” predstavljena tako, da bi na naslovu n bila koda črke “U” (85 ali 01010101 v dvojiškem številskem sestavu), na naslovu $n + 1$ bi bila koda črke “P” (80 oziroma 01010000) in na naslovu $n + 2$ koda črke “O” (79 oziroma 01001111).

Glede na to, da se danes večinoma uporablja pomnilniške medije, kjer je pomnilniška beseda dolga 8 bitov, se besedilo shranjuje na diskete, diske, zgoščenke in podobno, tako, da se za vsako črko shrani njena 7-bitna koda ASCII z dodanim bitom 0. Datoteke, ki vsebujejo samo 7-bitne kode ASCII, običajno imenujemo tekstovne ali datoteke z besedilom oziroma datoteke ASCII. Zanje je značilno tudi to, da če podatke iz take datoteke beremo v skupinah po 8 bitov, potem ima najpomembnejši bit take skupine bitov vedno vrednost 0. Datoteke, za katere to ne velja,

imenujemo netekstovne oziroma binarne.

Ker so kode 7-bitne kodne tabele po standardu ASCII zapisane s 7 biti, lahko z njimi predstavimo največ 128 (2^7) različnih znakov, kar za angleški jezik povsem zadostuje. Težave so se pojavile pri jezikih, ki uporabljajo še kakšne druge črke, na primer, v slovenščini uporabljamo šumnike, ki jih v 7-bitni tabeli ASCII ni. Zato so se izdelovalci programske opreme poslužili 8-bitnih tabel ASCII, kjer prvih 128 kod predstavlja enake znake kot 7-bitna tabela ASCII, medtem ko preostale kode predstavljajo neangleške znake. Pri 8-bitnih tabelah ASCII pa ni prišlo do enotne standardizacije predvsem zato, ker je vseh neangleških znakov v posameznih abecedah svetovnih znakov preprosto preveč. Samo pri pisavah, kot je kitajska, je precej več kot 256 znakov, zato je za predstavitev vseh njenih znakov ne zadostuje niti 8 bitov. Zato se dandanes uvaja kodna tabela Unicode, ki je 16-bitna, v novejših različicah tudi 32-bitna in s katero lahko trenutno opišemo 63468 oziroma 1048544 znakov, kar zadostuje za vse svetovne pisave skupaj.

Nekateri urejevalniki besedila, kot je npr. Microsoftov Word, shranjujejo besedilo v binarno obliko, tudi če gre za besedilo v angleškem jeziku, kjer se da vse črke predstaviti z 7 bitno tabelo ASCII oziroma v tekstovni obliki. Razlog leži predvsem v tem, da ti urejevalniki besedila omogočajo shranjevanje formatiziranega besedila. To pomeni, da poleg samega besedila shranjujejo še obliko pisave, velikost in barvo črk, barvo ozadja, razne slike, ipd.

Predstavitev slik

Slike so v računalniškem pomnilniku predstavljene s poljem slikovnih točk (angl. pixel). Za vsako točko na sliki je podana koda barve, v kateri naj se točka nariše. S tako predstavitvijo je sliko namreč najlažje narisati na zaslon. Točke si najlažje predstavljamo kot najmanjše enote slike, ki se vedno izrišejo v eni sami barvi. Pri sivinskih slikah potrebujemo za vsako točko nivo sivine, v kateri naj se ta točka izriše. Za predstavitev nivojev sivine se ponavadi uporablja 8 bitov, s katerimi lahko predstavimo 256 različnih nivojev sivine, pri čemer nivo 0 pomeni črno barvo, nivo 255 belo, vsi ostali pa vmesne sivine. Nivoji sivine so v pomnilniku zapisani kot nepredznačena cela števila. Sivinska slika velikosti $w \times h$ točk je v pomnilniku z dolžino pomnilniške besede 8 bitov predstavljena tako, da so sivine točk v prvi vrstici slike shranjene na pomnilniških lokacijah z naslovi od n do $n + w - 1$, sivine točk v drugi vrstici slike na pomnilniških lokacijah z naslovi od $n + w$ do $n + 2w - 1$, itd.

Na sliki 1.3 vidimo povečano sliko ikone iz enega izmed programov, v tabeli 1.3 pa vidimo vrednosti, na nekaterih pomnilniških lokacijah, če je ta črno-bela slika shranjena v pomnilniku z začetkom na pomnilniški

lokaciji z naslovom n. Velikost slike ikone je 17×18, torej se sivine točk v prvi vrstici slike nahajajo na pomnilniških lokacijah z naslovi od n do $n+16$, sivine točk v drugi vrstici slike pa na naslovih od $n+17$ do $n+33$, itd.

Slika 1.3: Slika ikone iz nekega programa

1. vrstica	*pomnilniške lokacije na naslovih od* n *do* $n+16$																
vrednost	191	191	191	191	191	191	4	4	4	4	4	4	4	4	191	191	191
2.vrstica	*pomnilniške lokacije na naslovih od* $n+17$ *do* $n+33$																
vrednost	191	191	191	191	191	191	4	242	242	242	242	4	242	242	4	191	191
3.vrstica	*pomnilniške lokacije na naslovih od* $n+34$ *do* $n+50$																
vrednost	191	191	191	191	191	191	4	242	4	242	4	4	242	242	242	4	191

Tabela 1.3: Predstavitev slike 1.3 v pomnilniku z začetkom na naslovu n

Barvne slike so v računalniškem pomnilniku običajno predstavljene tako, da je za vsako točko na sliki podana barva v obliki komponent RGB (angl. *Red, Green, Blue*). Komponente RGB so tri vrednosti, ki predstavljajo intenziteto rdeče, zelene in modre barve v barvi, ki jo predstavljajo. Vsaka izmed teh treh vrednosti je ponavadi predstavljena z 8 biti, kar pomeni, da lahko na tak način dosežemo 2^{24} različnih barv. Barvne slike zahtevajo tri take tabele, kot je tabela 1.3 (eno za vsako komponento), zato zasedejo trikrat toliko pomnilnika kot enako velike črno-bele slike. To velja seveda samo za predstavitev slik v pomnilniku, medtem ko so slike lahko na disku in ostalih zunanjih pomnilniških medijih zapisane v različnih zapisih, v katerih lahko ista slika zasede različno velik del prostora.

Predstavitev zvoka

Zvok je v računalniku predstavljen z zaporedjem vrednosti zvočnega signala, vzorčenega v različnih časovnih trenutkih. Za dobro kvaliteto zvoka je potrebno za vsako sekundo zvoka imeti vsaj 44100 vrednosti zvočnega signala, vzorčenih v enakomernih časovnih intervalih. Ta številka izhaja iz lastnosti človeškega sluha. Za predstavitev vrednosti zvočnega signala pa je ponavadi dovolj 16 bitov. To omogoča predstavitev 65536 različnih vrednosti, kar zagotavlja kvaliteten zvok. Podobno kot pri

predstavitvi besedila, so tudi pri zvoku vrednosti odčitane v sosednih intervalih zapisane v sosednih pomnilniških lokacijah. Začetek pesmi "Crazy" skupine Aerosmith je tako v računalniškem pomnilniku predstavljen z vrednostmi 185, 165, 547, 460, 1111, 928, 1480, 1358 itd., v binarni obliki. Tak način shranjevanja je primeren za predvajanje zvoka, vendar pa zasede veliko pomnilnika, saj v pomnilniku z dolžino pomnilniške besede 8 bitov, vsaka sekunda zvoka zasede vsaj 88200 pomnilniških lokacij. Zato se zvok na diskih in ostalih zunanjih pomnilniških medijih običajno hrani v posebnih zapisih (denimo MP3), ki precej zmanjšajo velikost datotek.

Predstavitev videa

Video je zaporedje slik, zato ga tako lahko predstavimo tudi v računalniškem pomnilniku. Vendar pa za kvaliteten video posnetek potrebujemo vsaj 25 slik na sekundo, kar zahteva precejšnjo količino pomnilnika. Ker so zaporedne slike v videu velikokrat zelo podobne, je smiselno shraniti le prvo sliko, za vsako naslednjo pa le spremembe od prejšnje slike. Tak način shranjevanja videa na zunanje pomnilniške medije uporablja zapis MPEG.

1.4 Programska oprema

Uporabniška programska oprema predstavlja množico programov, ki se izvajajo na večnamenskih računalnikih in omogočajo uporabnikom, da hitreje, lažje in zanesljiveje opravijo določene naloge.

Za začetek razvoja uporabniške programske opreme bi lahko razglasili leto 1979, ko je bila izdelana aplikacija VisiCalc za računalnike Apple, ki je bila prva elektronska preglednica. Z pojavitvijo VisiCalca so Applovi računalniki dobili novo razsežnost, saj so postali resnično koristen pripomoček pri vodenju poslovanja. Nekako v istem času je na tržišče prišel tudi prvi komercialno uspešen urejevalnik besedil WordStar, namenjen pa je bil IBM PC kompatibilnim računalnikom.

Med tipično splošnonamensko uporabniško programsko opremo sodijo skupine programov, predstavljene v tabeli 1.4. Našteti so tudi tipični predstavniki za tri danes najbolj razširjene platforme: računalniki PC z operacijskim sistemom Windows, računalniki PC z operacijskim sistemom Linux, ter računalniki Macintosh.

Glavne lastnosti, ki jih mora imeti uporabniška programska oprema, so predvsem intuitivnost operacij, zanesljivost, hitrost, dobra dokumentacija, združljivost z sorodnimi programi, skratka vse, kar velja tudi za dober uporabniški vmesnik.

	operacijski sistem / platforma		
skupine orodij	*Windows*	*Linux*	*Macintosh*
urejevalniki besedil	MS Word OpenOffice Writer Wordperfect TEX	OpenOffice Writer KWord AbiWord TEX	MS Word AppleWorks TEX
preglednice	MS Excel QuattroPro OpenOffice Calc	OpenOffice Calc KSpread	MS Excel AppleWorks
elektronske predstavitve	MS Powerpoint OpenOffice Impress	OpenOffice Impress KPresenter	MS Powerpoint AppleWorks
delo z bitnimi slikami	PhotoPaint Adobe Photoshop PhotoPlus	Gimp Krita	AppleWorks Adobe Photoshop
delo z vektorskimi slikami	Adobe Illustrator OpenOffice Draw CorelDraw	OpenOffice Draw Illustrator	Adobe Illustrator
odjemalci za elektronsko pošto	MS Outlook Eureka Calypso	KMail Pine Elm	Macintosh Mail
matematični programi	Mathematica Matlab Derive	Mathematica Matlab Octave	Mathematica

Tabela 1.4: Pregled najbolj uporabljene splošno namenske programske opreme na treh najbolj razširjenih platformah.

1.5 Naloge

1. Kako je v 8 bitnem registru nekega računalnika predstavljeno število 112?

2. Kako je v 16 bitnem registru nekega računalnika predstavljeno število -1237, če se za predstavitev negativnih števil uporablja eniški komplement?

3. Katero število predstavlja zapis 10110010 v primeru, da gre za zapis nepredznačenega števila in katero v primeru, da gre za zapis predznačenega števila v dvojiškem komplementu?

4. Kako je predstavljeno število 15.63 v zapisu s fiksno vejico, kjer je 8 bitov namenjenih za celi del, 8 pa za decimalni?

5. Kako je predstavljeno število -0.219 v zapisu s plavajočo vejico, kjer je 1 bit namenjen za predznak, 7 za eksponent in 8 za mantiso?

6. Število 18.253 zapišite v zapisu s fiksno vejico, kjer je 6 bitov namenjenih za celi del, 8 pa za decimalni. Ta zapis potem pretvorite nazaj v desetiški zapis. Kolikšna je razlika med številom, ki ste ga dobili in številom 18.253 (to razliko imenujemo tudi absolutna napaka pri zapisu števila v določenem zapisu)?

7. Ponovite prejšnjo nalogo, le da namesto zapisa s fiksno vejico uporabite zapis s plavajočo vejico, kjer je 1 bit namenjen za predznak, 5 za eksponent in 8 za mantiso.

8. Kateri sta najmanjše in največje pozitivno število, ki jih lahko povsem natančno predstavimo s zapisom v plavajoči vejici iz prejšnje naloge?

9. Ugotovite, najmanj koliko mest mora imeti zapis s plavajočo vejico namenjenih za mantiso, da je število 0.59 predstavljeno z napako, manjšo od 0.001?

10. Kako je v računalniškem pomnilniku z 8 bitnimi besedami tekst "LINUX" predstavljen s kodami po standardu ASCII?

1.6 Koristne spletne povezave

1. Zgodovina računalnikov podjetja Apple:
   `http://www.apple-history.com/`

2. Informacije o zapisih s plavajočo vejico po standardu IEEE 754:
   `http://osr5doc.sco.com:457/cgi-bin/printchapter/topics/BOOKCHAPTER-1.html`

3. Kratka zgodovina programov za urejanje besedil:
   `http://www.hodgy.net/computer_history/page_4/page_4 word processors.htm`

4. Kratka zgodovina elektronskih preglednic:
   `http://inventors.about.com/science/inventors/library/weekly/aa010199.htm`

5. Povzetek pomembnih dogodkov v zgodovini računalništva:
   `http://www.elsop.com/wrc/h_comput.htm`

1.7 Literatura

[1] D. Trček. *Informatika, Od tehnologije do poslovanja.* Visoka šola za management v Kopru, Koper, 2001.

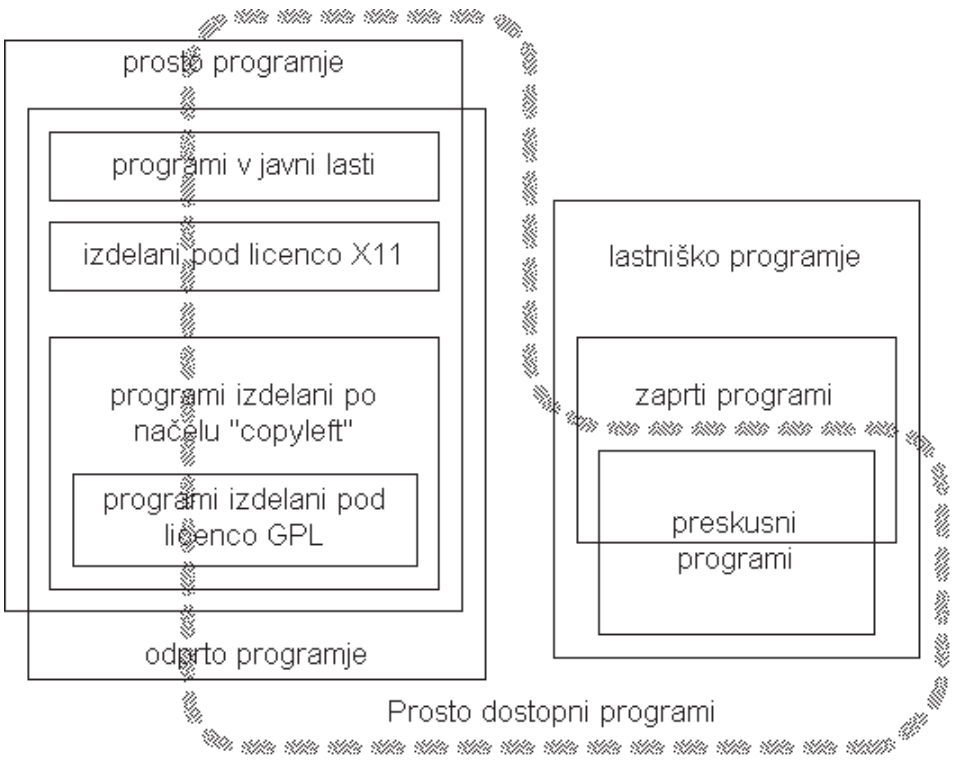
[2] D. Kodek. *Arhitektura računalniških sistemov.* Bi-Tim, Ljubljana, 2000.

[3] S. McCartney. *ENIAC The Triumphs and Tragedies of the Worlds's First Computer.* Walker and Company, New York, 1999.

[4] T. Zrimec. *Računalništvo in uporabniška programska oprema.* Fakulteta za računalništvo in informatiko, Univerza v Ljubljani, Ljubljana, 1997.

2

Pravna zaščita programske opreme



Slika 2.1: Nekatere kategorije prostega in neprostega programja

Programska oprema je lahko zakonsko zaščitena na različne načine. Licenca, ki jo uporabimo pri zaščiti, je odvisna od stopnje in namena, s katerim bi radi zaščitili program. Glede na uporabljene licence lahko programsko opremo v grobem delimo na prosto (angl. *Free Software Licences*) in neprosto (angl. *Non-Free Software Licences*) oz. lastniško

programsko opremo. Nekatere kategorije prostega in neprostega programja ter njihove medsebojne relacije prikazuje diagram Chao-Kueia. Posamezne kategorije bomo nekoliko natančneje opisali v nadaljevanju.

2.1 Prosto in odprto programje

Pripadniki prostega in odprtega programja se zavzemajo za nekatere svoboščine, ki naj bi pripadale uporabnikom in razvijalcem programske opreme. Razlike med obema skupinama so predvsem ideološke. Medtem ko pristaši prostega programja na prvem mestu zagovarjajo moralno-etično pravico za dostop do kode, so pripadniki odprtega programja osredotočeni bolj na kvaliteto kode, saj so prepričani, da dobimo najboljšo kodo, če vsem damo možnost, da prispevajo izboljšave.

2.1.1 Prosto programje

Za programsko opremo pravimo, da je prosta (angl. *free software*), če omogoča vse 4 prostosti, ki jih lahko podelimo uporabniku. Te prostosti se nanašajo na uporabo, proučevanje, spreminjanje in razširjanje. Prosta programska oprema torej daje uporabniku naslednje svoboščine:

prostost 0 : Program lahko svobodno poganjamo za kakršenkoli namen.

prostost 1 : Lahko proučujemo delovanje programa in ga prilagajamo svojim potrebam. Za to je seveda nujen dostop do izvorne kode (angl. *source code*).

prostost 2 : Program lahko prosto (brezplačno ali proti plačilu) razširjamo naprej.

prostost 3 : Program lahko izboljšujemo in po želji javno objavljamo izboljšane verzije za korist skupnosti.

Pri definiciji prostega programja je potrebno opozoriti, da prosto nikakor ne pomeni brezplačno. Prosto programje moramo razumeti v podobnem smislu kot svobodo govora in ne kot nekaj, kar je brezplačno ali zastonj, npr. brezplačno pivo. Prosto programsko opremo lahko distribuiramo brezplačno ali proti plačilu. Torej je lahko tudi prosta programska oprema komercialna. To lahko počnemo popolnoma svobodno. Nikogar nam ni potrebno vprašati za dovoljenje in nikogar ni potrebno o tem obveščati. Programe, ki so prosti, lahko po mili volji uporabljamo, spreminjamo in javno objavljamo. Enake pravice veljajo tudi za njihove spremenjene različice. Ker kodo lahko proučujemo in spreminjamo le s pomočjo izvorne kode, je seveda dostop do izvorne kode nujen pogoj, da za nek program lahko rečemo, da je prost.

Ohranjanje pravic prostega programja

Da pa prost program res tak tudi ostane, ga je potrebno zavarovati z licenco, ki te prostosti ohranja. V nasprotnem primeru bi se lahko zgodilo, da bi npr. nekdo vzel prost program in ga razširjal naprej kot lastniški program (angl. *proprietary software*). Programi, ki so prosti in niso zakonsko zaščiteni, spadajo v kategorijo **programja v javni lasti** (angl. *Public Domain Software*). Če je program v javni lasti, lahko kdorkoli napravi lastniško verzijo prej prostega programa.

Namen projekta GNU, ustanovljenega leta 1984 na pobudo Richarda Stallmana, je bil razviti operacijski sistem v stilu UNIXa, ki bi bil v celoti prost. Glavni razlog za ustanovitev omenjenega projekta je bil sociološko-etični, v smislu pozitivnega prispevka k skupnosti, saj bi bil omenjeni program na voljo vsem in brez omejitev. Za učinkovito širitev te ideje pa morajo biti proste tudi vse nadaljnje verzije programa. Metoda, ki te pravice ohranja, se imenuje **"Copyleft"** za razliko od "Copyright". Copyleft zakonsko odvzema pravico do prilaščanja programa in do spremembe njegovih uporabniških pravic ter daje bistvene pravice, določene s pojmom prostega programja, vsem uporabnikom. Poleg tega tudi prepoveduje kombiniranje prostih programov zaščitenih s copyleftom s programi, ki niso prosti, saj je taka kombinacija nujno neprost program. Copyleft določa, da mora biti vsaka kombinacija in nadgradnja programa zaščitenega s copyleftom še vedno prosta in zaščitena s copyleftom. Ko je program enkrat zaščiten po načelu copylefta, ostane prost za vedno.

Implementacija copylefta, ki je uporabljena v večini GNU-programov, se imenuje GNU GPL (GNU General Public Licence). Obstaja cel kup licenc za prosto programje, vendar so le nekatere med seboj združjive. Naj za primer omenimo licenco X11, ki ni licenca copyleft, je pa licenca za prosto programiraje in je združljiva z GNU GPL. Kdaj licenca določa prosto programje in katere licence so med seboj združljive, je včasih zelo težko določiti. Večkrat je potreben temeljit razmislek in posvetovanje z odvetnikom.

Podobno kot programsko kodo lahko zavarujemo tudi dokumentacijo. Copyleft za zaščito dokumentacije v okviru projeta GNU se imenuje GNU FDL (Free Documentation Licence) in daje uporabnikom podobne svoboščine kot GNU GPL.

Kako zaščitimo program z licenco GNU GPL?

Če ste torej razvili svoj program in želite, da bi se distribuiral kot prosta programska oprema z licenco GNU GPL, na začetek vsake datoteke z izvorno kodo dodajte naslednje vrstice:

V prvi vrstici navedite ime programa in kratek opis njegovega delovanja.

Copyright (C) *ime avtorja*

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

Na koncu pripišite tudi svoj kontaktni naslov: elektronski in/ali navadni naslov.

Na internetni strani `http://www.lugos.si/linux/gpl-sl.html` obstaja tudi neuraden poskus slovenskega prevoda licence GNU GPL. Z licencami, ki se nahajajo na omenjeni strani, ne morete uradno zaščititi svojih programov, pomagajo pa slovensko govorečemu prebivalstvu bolje razumeti licence GNU GPL.

Neuradni slovenski prevod GNU GPL se glasi takole:

Vrstica, v kateri podate ime programa in kratek opis, kaj počne. Copyright (C) 199l/2lll ime avtorja Ta program spada med prosto programje; lahko ga razširjate in/ali spreminjate pod pogoji Splošnega dovoljenja GNU (GNU General Public License), kot ga je objavila ustanova Free Software Foundation; bodisi različice 2 ali (po vaši izbiri) katerekoli poznejše različice.

Ta program se razširja v upanju, da bo uporaben, vendar BREZ VSAKRŠNEGA JAMSTVA; tudi brez posredne zagotovitve CENOVNE VREDNOSTI ali PRIMERNOSTI ZA DOLOČEN NAMEN. Za podrobnosti glejte besedilo GNU General Public License.

Skupaj s tem programom bi morali prejeti izvod Splošnega dovoljenja GNU (GNU General Public License); če ga niste, pišite na Free Software Foundation, Inc., Temple Place - Suite 330, Boston, MA 02111-1307, USA. Dodajte tudi informacije o tem, kako stopiti v stik z vami po elektronski ali papirni pošti.

2.1.2 Programiranje z odprto kodo

Zahteve odprtega programja (angl. Open Source) so podobne prostemu programju. Medtem ko je prosto programje mogoče nekoliko bolj razširjeno v raziskovalnih krogih in v krogih programerskih navdušencev, je odprto programje prodrlo tudi v poslovni svet. Ideja odprtega programja izhaja iz prostega programja. Vendar pa se pristaši odprtega programja pri razširjanju ideje opirajo na praktični ekonomski-poslovni vidik. Menijo namreč, da moramo dati ljudem možnost, da pregledajo in popravijo ter dopolnijo programe, saj to vodi h kvalitetnejši programski kodi. Pogoj za to je tudi v tem primeru dostop do izvorne kode.

Po definiciji odprtega programja so s pravnega stališča sprejemnljive tudi nekatere licence, ki za prosto programje niso. Licence odprtega programja ne zahtevajo distribucije identičnih ali spremenjenih kopij pod isto licenco, kot je določeno pri prostem programju, kjer se licenca prostega programja ohranja in je ni možno izničiti. To vodi v nekakšno nesimetrijo med prostim in odprtim programjem. Medtem ko je možno vgraditi kodo odprtega programja v program prostega programja, obratno ni možno. Prosto programje je popolnoma ločeno od lastniškega programja. Nobene pravne možnosti ni, da bi združevali kodi lastniškega in prostega programja. To vodi v nekakšno napetost med lastniškim in prostim programjem. Pristaši odprte kode so nekako manj zaprti in z veseljem sodelujejo tako s podjetji kot z gibanjem prostega programja.

Izraza odprto in prosto programje sta v zadnjem času precej popularna, kar izkoriščajo različna podjetja za komercialne namene. Na svoje izdelke zavajajoče nameščajo napise, ki vsebujejo besede “open source” ali “free software”. Napis, da je nekaj narejeno po vzoru odprtega programja, še ne pomeni, da je tudi sam produkt odprto programje. Pri takih stvareh moramo biti izjemno previdni, saj nas reklame lahko hitro zavedejo.

Programi, ki so na CD-ju priloženi tej knjigi, spadajo med prosto in odprto programsko opremo.

2.2 Lastniško programje

V nasprotju z opisano licenco GNU GPL, ki uporabnikom daje pravice, jih ostale avtorske licence odvzemajo. Lastniško programje (angl. *proprietary software*) prepoveduje oziroma omejuje uporabo, razširjanje in spreminjanje programov.

Pojma lastniško programje ne smemo zamenjevati s pojmom komercialno programje (angl. *commercial software*), čeprav ta dva pojma res največkrat srečujemo skupaj. Cilj komercialnega programja je zaslužek. Komercialni programi so lahko prosti ali neprosti. Prav tako obstaja tudi

nekomercialno programje, ki ni prosto.

2.2.1 Primer komercialnih licenc

Vsako podjetje ima svojo poslovno strategijo in temu prilagaja tudi licence, ki morajo slediti tehničnemu razvoju. Tako je naprimer z razmahom mrežnih povezav potrebno poskrbeti za pravni red tudi v medmrežju. Sledi splošen opis različnih tipov komercialnih licenc, ki je v večini povzet po Microsoftovih izdelkih.

Komercialna programska licenca

Komercialna programska licenca daje pravico do uporabe programskega izdelka in hkrati določa, na kakšen način lahko program uporabljamo in distribuiramo.

Preden kupimo programsko opremo, moramo najprej odgovoriti na vprašanja, kot so, koliko uporabnikov potrebuje določen programski izdelek, na koliko računalnikov ga bomo namestili, kako pogosto želimo verzijo programa nadgraditi, ali želimo o izdelku kakšno dodatno izobraževanje, kakšen bo način plačila itd. V nekaterih primerih pa programsko opremo že kupimo skupaj s strojno opremo, primer so IBM-ovi prenosniki, kjer poleg računalnika kupimo tudi Microsoftov operacijski sistem Windows.

Če bo programsko opremo uporabljalo le majhno število uporabnikov in bo ta nameščena na majhnem številu računalnikov, potem je verjetno najboljša izbira nakup posameznih programskih izdelkov skupaj z ustrezno licenco. Če pa kupujemo večje število, potem je vsekakor smiselno vprašati za ugodnosti, ki jih dobimo pri večjem nakupu. Ni nujno, da programsko opremo kupimo, obstajaja tudi možnost najema za določen čas.

Posebne ugodnosti programske hiše namenjajo tudi izobraževalnim, dobrodelnim in vladnim organizacijam. Tako je npr. Fakulteta za računalništvo in informatiko pristopila k programu Microsoft Developer Network Academic Alliance (MSDNAA), ki omogoča pridobitev Microsoftovih razvojnih orodij, platform in strežnikov za uporabo v izobraževalne in raziskovalne namene. Do uporabe programske opreme iz programa MSDNAA so upravičeni vsi redni študenti Fakultete za računalništvo in informatiko ter zaposleni na Fakulteti za računalništvo in informatiko. Vsi, ki so upravičeni do uporabe programske opreme iz programa MSDNAA, lahko dobijo programsko opremo iz programa MSDNAA tudi za uporabo na osebnem računalniku doma v pedagoške in/ali nekomercialne raziskovalne namene. Pri uporabi te programske opreme morajo ravnati v skladu z zahtevami Licenčne pogodbe za

končnega uporabnika programa MSDN in Dopolnila k navedeni licenčni pogodbi, da bi zadostili zahtevam programa MSDNAA.

2.2.2 Poskusni programi

Programi na pokušino (angl. *Shareware*) so delno ali popolnoma funkcionalni javno dostopni programi (ali podatki), ki jih lahko razširjamo naprej. Pri večini izvorna koda ni dostopna. Spreminjanje torej ni možno, navadno tudi ni dovoljeno. Njihova dolgotrajna uporaba je običajno dovoljena proti plačilu. Tudi podpora je možna le proti plačilu. Veliko ljudi take programe kljub neplačilu še vedno uporablja. Taka uporaba je sicer možna, če delovanje programa ni časovno omejeno, ni pa legalna.

2.2.3 Demonstracijske različice

Pojem demonstracijske različice (angl. *demo version*) ima dva pomena.

1. Zgodnja ne povsem funkcionalna vezija programa v razvoju. Take verzije imajo navadno izdelan uporabniški vmesnik, funkcije v ozadju pa niso popolnoma implementirane in navadno še vsebujejo programerske napake. Namenjene so predstavitvi naročniku oz. končnemu uporabniku ter reševanju morebitnih nesporazumov pri načrtovanju programa.

2. Posebna, navadno okrnjena verzija že končanega programa, ki jo lahko dobimo brezplačno ali pa za minimalno protiplačilo. Take verzije so namenjene potencialnim uporabnikom-kupcem programa, ki lahko pred nakupom doma sami preizkusijo program.

2.2.4 Poskusni programi

Poskusna različica (angl. *trial version*) programov lahko brezplačno namestimo na računalnik. Navadno vsebujejo celoten program in dokumentacijo. Njihovo delovanje je največkrat časovno omejeno na nekaj ur ali nekaj dni. Po izteku časovnega roka teh programov ni več možno uporabljati. Namenjeni so seznanjanju potencialnih uporabnikov s programom.

2.2.5 Zastonjsko programje

Zastonjsko programje (angl. *freeware*) se pogosto uporablja za pakete, ki dovoljujejo razširjanje, ne pa spreminjanja. Tudi njihova izvorna koda ni dostopna.

2.2.6 Programje podprto z oglaševanjem

Programi, ki prikazujejo reklamne oglase (angl. *adware*) so brezplačni programi. Finančno so podprti s pomočjo oglasov, ki so vključeni v program. Adware programi imajo navadno tudi svojo plačljivo različico, ki pa ne vsebuje komercialnih sporočil.

2.3 Avtorska zaščita programske opreme

Računalniški programi sodijo po slovenski in mednarodni zakonodaji med avtorska dela. V Slovenji varuje računalniške programe Zakon o avtorskih in sorodnih pravicah (Uradni list RS, št. 21/95, št. 9/2001, št. 43/2004), ki jih uvršča med pisana avtorska dela. Na splošno varujeta avtorsko pravico na računalniških programih 21. in 22. člen zakona, poleg njiju pa še posebni oddelek 4. poglavja zakona. Sicer pa zanje veljajo vsa pravila omenjenega zakona ter ratificirane mednarodne pogodbe, med katerimi sta najpomembnejši Bernska konvencija za varstvo književnih in umetniških del ter Sporazum o trgovinskih vidikih pravic intelektualne lastnine - TRIPS v okviru Svetovne trgovinske organizacije.

Programsko opremo obravnava:

- 60. člen Ustave Republike Slovenije
- Zakon o avtorskih in sorodnih pravicah (kratica ZASP), UL RS št. 21/95 s spremembami in z dopolnitvami, UL RS št. 9/2001, UL RS št. 43/2004
- Kazenski zakonik Republike Slovenije, UL RS št. 63/94 s spremembami in dopolnitvami, UL RS št. 23/99, UL RS št. 40/2004
- Zakon o preprečevanju omejevanja konkurence, UL RS št. 56/99
- Zakon o sodiščih, UL RS št. 19/94, 38/99 in 28/2000
- Zakon o dohodnini, UL RS št. 71/93 s spremembami in dopolnitvami, UL RS št. 7/95, 44/96

Sankcije za nelegalno uporabo so podrobneje definirane v Kazenskem zakoniku RS in v Zakonu o avtorski in sorodnih pravicah. Kazenski zakonik določa do 8 let zaporne kazni za nelegalno izkoriščanje računalniških programov. ZASP med drugim določa tudi kazen za nelegalno uporabo avtorsko zaščitenih del v znesku od 400.000 do 10.000.000 SIT in pooblašča tržno inšpekcijo za nadzor nad legalno uporabo.

2.4 Kaj je piratstvo?

Obstaja pet osnovnih oblik programskega piratstva:

1. **Ponarejanje** (angl. *Counterfeiting*), pri katerem gre za neavtorizirano reproduciranje in distribuiranje zavarovanih programov na disketah ali CD-ROM-ih v opremi (embalaži), ki je ponavadi tudi ponarejena;

2. **Nalaganje na disk** (angl. *Hard Disc Loading*), pri katerem gre za to, da proizvajalci ali prodajalci pri prodaji računalnikov brezplačno opremijo računalnik z neavtoriziranimi programi, s čimer želijo pridobiti kupce;

3. **Mehko piratstvo** (angl. *Softlifting*), pri katerem podjetje nabavi samo eno legalno kopijo programa in jo neavtorizirano reproducira na vse svoje računalnike oziroma svoje delavce;

4. **Dajanje v najem** (angl. *Software Rental*) primerkov avtorskega dela, npr. prodajalec posodi programsko opremo za uporabo na izposojevalčevem domačem ali službenem računalniku;

5. **Internetno piratstvo**, kjer gre za neavtorizirano naložitev računalniškega programa na spletno stran oziroma nezakonito razpečavanje s pomočjo interneta (angl. slengovski izraz za piratske programe, filme ipd. je *Warez*).

2.5 Nadzor programske opreme v Sloveniji

BSA (*Business Software Alliance*) je mednarodna organizacija, glasnik vodilnih proizvajalcev in prodajalcev programske in strojne opreme ter ostalih podjetij, ki s svojim delovanjem bistveno pripomorejo k razvoju interneta in elektronskega poslovanja. Več let deluje tudi v Sloveniji. Njen glavni namen je obveščanje in izobraževanje javnosti o upravljanju s pravicami na računalniških programih, avtorsko pravno zaščito, zaščito v kibernetskem prostoru, elektronskem poslovanju in z internetom povezanimi vprašanji ter podpora organom pregona pri aktivnostih za zatiranje nedovoljene reprodukcije, distribucije in uporabe nelicenčnih računalniških programov.

Aktivno se v boj proti piratstvu vključuje tudi Tržni inšpektorat Slovenije. V skladu z Zakonom o avtorskih in sorodnih pravicah (členi od 184 do 186) lahko tržna inšpekcija brez naloga sodišča preišče podjetja, osumljena kršitev avtorskih pravic. Na Tržnem inšpektoratu Slovenije so ustanovili posebno skupino desetih inšpektorjev iz glavnih slovenskih

regij, ki so specializirani za postopke proti računalniškim piratom. Postopki naj bi bili tako še preprostejši in učinkovitejši.

Stopnja piratstva se v Sloveniji zmanjšuje. Tako je npr. leta 1994 znašala 96%, v letu 2002 pa 59%. Slovenija je danes država z najnižjo stopnjo piratstva med državami vzhodne Evrope. K zmanjševanju prispevajo v veliki meri tudi pogosti postopki pred sodniki za prekrške. V zadnjih letih je bilo opravljeno kar nekaj pregledov programske opreme pri končnih uporabnikih. Uporabniki nelegalne programske opreme morajo plačati precej visoke denarne kazni. Najvišja kazen, ki so jo doslej izrekli, je bila za pravno osebo[1] 3.000.000,00 SIT, za fizično osebo[2] pa 400.000 SIT.

2.6 Avtorske pravice

Katere pravice pripadajo kot avtorjem programske opreme, torej programerjem? Ker računalniški programi sodijo med avtorska dela, programerejm pripadajo vse pravice iz vsebine avtorskega prava (ZASP). To so moralne avtorske pravice, materialne avtorske pravice in druge pravice avtorja.

2.6.1 Zakonsko določene avtorske pravice

Moralne avtorske pravice varujejo avtorja glede njegovih osebnih vezi do dela. Tako ima avtor pravico določiti, kje in kdaj bo njegovo delo privič objavljeno. Avtor ima pravico določiti, s kakšno oznako naj se navede njegovo avtorstvo, pravico, da se upre skazitvi ali kakšnemu drugemu posegu v njegovo delo. Prav tako ima avtor pod posebnimi pogoji pravico do vrnitve materialnih pravic, če ima za to resne moralne razloge.

Materialne avtorske pravice varujejo premoženjske interese avtorja s tem, da avtor izključno dovoljuje ali prepoveduje uporabo svojega dela. Če ni z zakonom ZASP drugače določeno, je uporaba avtorskega dela dopustna le, če je avtor v skladu s tem zakonom in pod pogoji, ki jih je določil, prenesel ustrezno materialno avtorsko pravico. Materialne pravice vsebujejo pravico reproduciranja, distribuiranja, dajanja v najem, javnega izvajanja, javnega prenašanja, javnega prikazovanja, predelave idr.

ZASP ima tudi posebej oddelek namenjen izključno računalniškim programom (četrto poglevje, 2. oddelek). Tu je definiran pojem računalniškega programa in njegovega varstva na naslednji način:

1. Računalniški programi po tem zakonu so programi v vsaki izrazni

[1] pravna oseba: pravno priznana skupnost z namenskim premoženjem kot nosilec pravic in dolžnosti (delniška družba d.d.; družba z omejeno odgovornostjo d.o.o.; družba z neomejeno odgovornostjo d.n.o.; samostojni podjetnik s.p.);

[2] fizična oseba: posameznik.

obliki, vključno s pripravljalnim gradivom za njihovo izdelavo.

2. Ideje in načela, ki so osnova nekemu elementu računalniškega programa, vključno s tistimi, ki so osnova njegovim vmesnikom, ne uživajo varstva.

3. Računalniški programi uživajo varstvo, če so individualna dela v tem smislu, da pomenijo lastno intelektualno stvaritev njihovega avtorja.

Ta del zakona med drugim določa tudi pravice delodajalca in avtorja programske opreme, vsebinske omejitve avtorjevih pravic, dekompiliranje ali razgradnja programov ter posebno varstvo računalniških programov.

Tako določa, da kadar računalniški program ustvari delojemalec pri izpolnjevanju svojih obveznosti ali po navodilih delodajalca, ali ga ustvari avtor po avtorski pogodbi o naročilu, se šteje, da so materialne avtorske pravice in druge pravice avtorja na tem programu izključno in neomejeno prenesene na delodajalca ali naročnika, če ni s pogodbo drugače določeno.

Poznavanje vseh pravic in dolžnosti v okviru avtorskega prava vsekakor zahteva natančno študijo pravnih virov, ki te stvari urejajo.

2.6.2 Avtorska agencija za Slovenijo

Ker je varovanje lastnih avtorskih pravic ob nepoznavanju in nespremljanju zakonodaje, težka naloga, se avtorji (tudi programerji) v Sloveniji lahko obrnejo na Avtorsko agencijo za Slovenijo (AAS). AAS se ukvarja s pravnim zastopanjem avtorjev in imetnikov avtorskih pravic pri uveljavljanju in varstvu njihovih pravic, s pravnim svetovanjem na področju avtorskega prava, z izdelavo neodvisnih izvedenskih mnenj s področja avtorske in sorodnih pravic ter z registracijo avtorskih del.

2.7 Programski patenti

Programski patenti so zadnjem času zelo vroča tema v Evropskem parlamentu. Patent definiramo kot pogodbo med izumiteljem in javnostjo. Izumitelj v zameno za časovno in prostorsko omejen monopol objavi svoj izum. Na ta način naj bi se vzpodbujala izumiteljska dejavnost, saj z objavo svojega znanja omogoči drugim, da to razvijajo in izboljšujejo naprej. Za programske patente se zavzemajo velika podjetja, ki si patente lahko privoščijo. Patentiranje namreč ni poceni, se pa splača, saj kasneje lahko prinaša velike dobičke. Pristaši odprtega programja zelo intenzivno nasprotujejo programskim patentom. Trdijo, da bi patenti preprečili zdrav konkurenčni razvoj programja. Nekateri scenariji, ki jih navajajo kot možne posledice legaliziranja programskih patentov, so:

Evropski patentni urad (angl. European Patent Office, EPO) je že podelil več kot 30.000 programskih patentov, ki pa jih zaenkrat še ni mogoče uveljaviti. Že podeljeni patenti vsebujejo tudi operator XOR, prikaz poteka (angl. progress bar), uporaba t.i. null operatorjev za upočasnitev dogajanja, uporaba dveh barv za prikaz popravkov v dokumentu, drsni trak (angl. scroll bar), uporaba miškinega klika za nakupovanje preko spleta itd. Vse te metode so vključene v velik del že obstoječe odprte programske kode. Legalizacija takih programskih patentov bi ogrozila obstoj odprtega programja, saj je uporabo patentov potrebno plačati. Posledica programskih patentov v ZDA se kaže tudi v skrajnostih, kot je npr. primer formatov MP3 in GIF. V ZDA je zakonsko prepovedano pisanje komercialnih programov, ki bi shranjevali zvok v format MP3 oz. sliko v format GIF (oziroma je bilo, dokler je veljal patent) brez plačila ustrezne licenčnine.

Tako pristaši programskih patentov kot njihovi nasprotniki imajo svoje argumente in interese. Kako se bo stvar razpletla, pa bomo še videli.

2.8 Internetno pravo

Značaj interneta je odprl kup pravnih vprašanj, ki so po večini še vedno nerešena. Internet namreč omogoča objavljanje poljubnih vsebin in izmenjavo datotek, med katerimi pa so tudi avtorska dela. Eden izmed najbolj odmevnih primerov kršenja avtorskih pravic je tožba proti Napsterju. Program Napster je omogočal brezplačno izmenjavo datotek med poljubnima računalnikoma povezanima v svetovni splet. Sistem je deloval na podlagi "jaz tebi – ti meni"(angl. P2P – Point to Point), pri katerem vsak uporabnik določi, katere datoteke je pripravljen deliti z drugimi. Program je enostaven za uporabo in je bil v času svojega delovanja izredno popularen. Ker tak sistem omogoča hitro širjenje datotek, se je izkazal kot idealen za širjenje datotek MP3, torej glasbe, je upadla prodaja originalnih avtorskih del na plačljivih CD-jih. Ameriško združenje za avtorske pravice RIAA je zato sprožilo sodni proces zoper Napster. Ker je podjetje vedelo, da se njihov program uporablja zgolj za nelegalne namene, torej za izmenjevanje avtorske glasbe, je sodišče delovanje Napsterja prepovedalo. Poleg glasbe internet ogroža vse izdelke, ki se lahko pretvorijo v digitalno obliko. Na udaru je filmska industrija, knjižne založbe, proizvajalci programske opreme itd.

2.9 Naloge

1. Kdo v Sloveniji skrbi za zmanjševanje piratstva?
2. Kakšna je bistvena razlika med prosto in odprto programsko

opremo?

3. Ali obstaja možnost, da prosto programsko opremo spremenimo v avtorsko?

4. Zakaj preskusnih programov ne moremo uvrstiti med prosto programsko opremo?

5. Kakšno vrsto licence moramo uporabiti, če združimo prosto programsko opremo in odprto programsko opremo v skupen izdelek?

6. Kakšna je kazen, če CD, priložen tej knjigi, 10x skopiraš in prodaš svojim znancem?

7. Ali so preskusni programi (angl. *shareware*), ki jih lahko poljubno razširjamo, odprto programje?

2.10 Koristne spletne povezave

1. Slovenska spletna stran BSA (*Business Software Alliance*):
   `http://www.bsa.si/`

2. Neuraden poskus slovenskega prevoda licence GNU GPL:
   `http://www.lugos.si/linux/gpl-sl.html`

3. GNU Project:
   `http://www.gnu.org/`
   `http://gnu.fyxm.net/philosophy/categories.sl.html`

4. The Free Dictionary online:
   `http://computing-dictionary.thefreedictionary.com/`

5. Microsoftove licence:
   `http://www.microsoft.com/licensing/default.mspx/`

6. Internet in pravo:
   `http://pf-lj.kelt.si/internetinpravo/index.htm`

7. Avtorska agencija za Slovenijo (AAS):
   `http://www.aas.si/index.htm`

2.11 Literatura

[1] J. Rosenoer. *CyberLaw, The Law of the Internet.* Springer, New York, 1997.

3

Uporabniški vmesniki

V kratki zgodovini razvoja računalnikov se je način komuniciranja uporabnikov z računalniki spreminjal in se stalno izpopolnjeval. Prve računalnike so programirali s pomočjo stikal in celo s pretikanjem električnih kablov, računalnik pa je rezultate oziroma potek procesiranja javljal z utripajočimi lučkami. Kot glavni medij za zapis programov in podatkov so se uporabljali luknjani papirnati trak in luknjane kartonske kartice. Tipkovnica je že zelo zgodaj postala najpomembnejša naprava za vnašanje informacij v računalnik. V povezavi z monitorjem je tipkovnica še danes osrednji del vsakega uporabniškega vmesnika.

Z razvojem grafičnih uporabniških vmesnikov se je kot nepogrešljiv element uporabniškega vmesnika pojavila še računalniška miška, s katero lahko izbiramo in premikamo objekte na računalniškem zaslonu. Prvi prototip miške je leta 1968 predstavil Douglas Englebart.

Uporabniški vmesniki pa se razvijajo naprej. Poseben izziv predstavljajo nove vrste prenosnih računalnikov, kjer klasične tipkovnice zaradi njene velikosti ne moremo uporabiti. Tako postaja pri komuniciranju z računalnikom poleg zaslonov na dotik in raznih vrst elektronskih peres vse bolj aktualna uporaba govora in vizualnih informacij. Uporabnik lahko ukazuje računalniku z besedami, ki jih mora računalnik prepoznati, na drugi strani pa lahko tudi računalnik svoje rezultate uporabniku sporoča s sintetiziranim govorom. Podobno lahko računalnik s pomočjo video kamer in drugih senzorjev zaznava ožjo in širšo okolico in razpoznava akcije in kretnje uporabnika, rezultate procesiranja pa uporabnik lahko vidi kot grafične upodobitve na računalniškem zaslonu ali pa, s pomočjo posebnih naprav, celo v obliki 3-dimenzionalnih virtualnih objektov.

V preteklosti so z računalniki rokovali posebej visoko izobraženi strokovnjaki. Danes pa mora z računalniki delati vedno širši krog ljudi. Zato morajo biti uporabniški vmesniki veliko bolj prilagodljivi različnim vrstam uporabnikov. Uporabnike računalniških aplikacij običajno

razvrstimo v tri skupine:

začetnike, ki nimajo nič računalniškega predznanja in jih je običajno strah, ko se prvič soočijo z računalnikom;

občasne uporabnike, ki sicer razumejo osnovne računalniške koncepte, vendar nimajo konkretnega znanja o sintaksi ukazov v posameznih uporabniških programih; in

pogoste uporabnike, ki stalno delajo z računalniki in zahtevajo predvsem hitro izvedbo svojih ukazov.

Glede na vrsto uporabnikov so bolj ali manj pomembne tudi različne vrste komuniciranja z računalnikom. Za začetnike in občasne uporabnike je primerno predvsem komuniciranje preko grafičnega vmesnika, kjer lahko uporabnik izbira med več možnimi ukazi s pomočjo miške in manipulira neposredno z objekti na zaslonu (datoteko na primer zbriše tako, da jo prenese na ikono, ki predstavlja koš za smeti). Tak način dela tudi ne zahteva predhodnega dolgega učenja uporabe nove programske opreme.

Čeprav je taka interakcija zelo intuitivna, pa je za pogoste uporabnike lahko enostavno prezamudna. Zato je za take uporabnike in predvsem za tiste programe, s katerimi delajo večino svojega delovnega časa, veliko bolj primerno vnašanje ukazov preko tipkovnice. Za načrtovalce uporabniških vmesnikov predstavlja zato poseben izziv zgraditi tak uporabniški vmesnik, ki se lahko prilagaja različnim vrstam uporabnikov.

3.1 Lastnosti in ocenjevanje uporabniških vmesnikov

Pri obravnavi uporabniških vmesnikov ločimo štiri različne *abstrakcijske nivoje*:

konceptualni nivo, ki določa osnovni način interakcije in druge osnovne pojme,

semantični nivo, ki določa pomen posameznih ukazov, vhodne in izhodne enote ter katere informacije so potrebne za določeno akcijo;

sintaktični nivo, ki določa specifični vrstni red argumentov v ukazih in pravila za urejenost enot na vhodu in izhodu; in

leksikalni nivo, ki govori o podrobnostih, ki so odvisne od naprav, in o točnih mehanizmih, ki določajo sintakso.

Pri učenju in pomnjenju ukazov za delo z računalnikom moramo upoštevati, da si veliko lažje zapomnimo način dela na konceptualnem in

semantičnem nivoju kot pa sintaktične ali celo leksikalne podrobnosti. Prav zato, ker ima večina sodobne aplikacijske programske opreme podoben konceptualni in v veliki meri tudi semantični nivo, lahko uporabniki že z malo truda hitro zamenjajo svoje programsko okolje.

Dober uporabniški vmesnik naj bi izpolnjeval vse naslednje zahteve:

popolnost: uporabnik lahko izrazi vse ukaze,

skladnost: v podobnih situacijah naj se uporablja podobna zaporedja akcij,

učinkovitost: hitra izvedba s čim manj možnimi napakami (možnost definiranja bližnjic),

razširljivost: dodajanje novih pojmov s pomočjo obstoječih,

odzivnost: računalnik se mora ustrezno odzvati na uporabnikov ukaz,

možnost popravljanja napak: obrnljivost akcij, detekcija napak,

nudenje pomoči: enostavna sporočila na zaslonu, na zahtevo daljša razlaga, predvsem o sintaksi vhodnih ukazov.

Pri ocenjevanju in primerjavi različnih uporabniških vmesnikov moramo upoštevati naslednje faktorje:

1. čas učenja,
2. hitrost izvedbe značilnih ukazov,
3. možne napake uporabnikov,
4. subjektivno zadovoljstvo uporabnikov,
5. pomnjenje,
6. ceno vmesnika oziroma programske rešitve.

Z vidika ekonomičnosti se zato pogosto izkaže, da so po nabavni ceni sicer dražje programske rešitve na dolgi rok cenejše, saj se jih uporabniki hitreje naučijo in z njimi lahko tudi hitreje in bolj učinkovito delajo.

3.2 Načini interakcije z računalnikom

Glede na način interakcije lahko uporabniške vmesnike razdelimo na dve veliki skupini: tiste, ki uporabljajo ukazni jezik, za kar sta potrebna le tipkovnica in zaslon, ter tiste, ki uporabljajo poleg tipkovnice in zaslona še računalniško miško, kar omogoča lažjo izbiro med več ponujenimi možnostmi na zaslonu ter direktno manipulacijo objektov, ki so prikazani na zaslonu.

3.2.1 Ukazni jezik

Zahtevana oprema: tipkovnica in računalniški zaslon.

Način dela: uporabnik tipka vse ukaze.

Prednosti:

- hitrost (pomembna za pogoste uporabnike!),
- fleksibilnost (možnost tvorjenja makro ukazov),
- najmanj zahtev glede strojne in programske opreme.

Slabosti:

- dolgo šolanje,
- veliko pomnjenja in zato hitro pozabljanje ukazov.

Primeri:

- operacijska sistema DOS in Unix,
- urejevalnik besedila `vi` na Unixu.

Delo s pomočjo ukaznih jezikov je še vedno zelo razširjeno zato, ker je tipkanje ukazov, sicer s potrebno vajo, še vedno najhitrejši način komuniciranja z računalnikom.

3.2.2 Izpolnjevanje formularjev

Zahtevana oprema: tipkovnica in računalniški zaslon.

Način dela: uporabnik s tipkanjem vnaša podatke v vnaprej pripravljena polja. Med polji se premika s pomočjo tipke `Enter>`.

Prednosti:

- poenostavljen vnos podatkov,
- posnemanje formularja na papirju,
- kratko šolanje.

Slabosti:

- namenjeno specifični, ponavljajoči se nalogi (npr. kadrovska evidenca),
- požrešno glede prostora na zaslonu.

Primeri:

- različni uporabniški programi,
- izpopolnjena oblika formularjev so **preglednice**, kjer imajo celice več funkcionalnosti (matematične in logične operacije nad več celicami). Dodani so elementi direktne manipulacije. Preglednice obravnavamo v 4. poglavju.

3.2.3 Izbira preko menijev

Zahtevana oprema: tipkovnica, računalniški zaslon in miška.

Način dela: uporabnik s pomočjo miške ali tipk izbira med ponujenimi ukazi.

Prednosti:

- kratko šolanje (namesto pomnjenja ukazov je potrebno le njihovo prepoznavanje),
- strukturirano (hierarhično) odločanje.

Slabosti:

- zavzame veliko prostora in včasih oteži izbiro zaradi prevelikega števila ukazov. Zato se ukaze lahko strukturira in grupira s pomočjo izvlečnih in dvižnih menijev[1]. Ukaze v posameznem meniju lahko razvrstimo po abecedi, po kategoriji, po vrsti ali po pogostnosti uporabe. Ukazi so lahko v meniju razen po linearnem načinu razporejeni še v krogu ali v matriki.
- upočasnuje pogoste uporabnike.

Primeri:

- bančni avtomati,
- okenski uporabniški vmesniki,
- mobilni telefoni.

3.2.4 Neposredna manipulacija

Zahtevana oprema: tipkovnica, zaslon (stereo očala, čelada VR), miška (podatkovna rokavica).

[1] V slovenščini se za meni uporablja tudi izraz izbirnik.

Način dela: uporabnik s pomočjo miške ali drugih interaktivnih vmesnikov izbira objekte na zaslonu (ali v virtualnem prostoru) in z njimi upravlja ali manipulira.

Za neposredno manipulacijo je potrebno vizualno predstaviti vse objekte. Okolje, v katero so objekti postavljeni, imenujemo namizje; datoteke, imeniki in posamezni programi pa so na namizju predstavljeni z ikonami. Ikone naj bi bile oblikovane tako, da jih lahko hitro ločimo med seboj in da že po obliki ikone vidimo, s katerim programom je bila datoteka ustvarjena. S pomočjo miške lahko objekte izbiramo in neposredno manipuliramo (premikamo – akcija povleci in spusti, odpiramo itd.). Pri neposredni manipulaciji so možne hitre, reverzibilne in inkrementalne akcije.

Prednosti:

- hitro učenje,
- uporabnik neposredno vidi, če akcije vodijo k cilju,
- uporabnik pridobi samozaupanje in nadzor.

Slabosti:

- zahtevna za programiranje,
- zmogljiva strojna oprema,
- vseh nalog ne moremo opisati s konkretnimi objekti in neposredno akcijo.

Primeri:

- Xerox Star,
- Apple Macintosh,
- Microsoft Windows,
- X Windows
- video igre.

Neposredna manipulacija se uporablja predvsem v okviru grafičnih uporabniških vmesnikih, ki imajo naslednje sestavne dele:

- namizje (angl. *desktop*),
- ikone (programi, datoteke, mape – direktoriji),
- okna (imenik ali področje izvajanja programa), drsniki za določanje izreza, in druge kontrole, ki omogočajo premikanje in spreminjanje velikosti oken,

- meniji (izvlečni, dvižni, kontekstni),
- miška, ki omogoča klik, dvojni klik, izbiro in premik,
- animacije.

3.3 Oblikovanje uporabniških vmesnikov

Pri uporabniški programski opremi ne moremo mimo uporabnika. Ta je končni razsodnik kakovosti. Uporabnik bo s programsko opremo zadovoljen le, če mu bo ta delo olajšala. Pod pojmom *uporabnik* bomo v nadaljevanju obravnavali zlasti netehničnega uporabnika, ki se na računalnike praviloma ne spozna najbolje. Izkaže se, da je tisto, kar je dobro zanj, pogosto dobro tudi za bolj tehničnega uporabnika. Prvi se ne zna, drugi pa noče ukvarjati s tehničnimi problemi, ki se neposredno ne tičejo naloge, zaradi katere uporabnik računalnik sploh uporablja. Nameščanje gonilnikov naprimer ni nikomur v veselje.

Izdelava uporabniku prijaznih programov je v resnici bolj predmet *oblikovanja* (industrijskega, ne zgolj vizualnega!), kot pa programiranja. Oblikovanje navadno bolj povezujemo z umetnostjo kot pa z natančnim racionalnim inženirstvom, kamor naj bi spadalo tudi računalništvo in programiranje. Oblikovanje se tudi tradicionalno poučuje na umetniških akademijah. Za razliko od nekaterih drugih umetniških praks, denimo slikarstva, pa imamo pri oblikovanju navadno podane bolj ali manj natačne *zahteve*, ki naj bi jim oblikovani izdelek zadoščal. Oblikovani stol je lahko še tako lep, vendar ga ne moremo imeti za uspešen oblikovalski dosežek, če na njem ne moremo udobno sedeti.

Oblikovanje je pravzaprav sklepanje kompromisov. Poglejmo naprimer koš za smeti. Biti mora 1) velik, da vanj spravimo veliko smeti in 2) majhen, da ne zavzame preveč prostora. Biti mora 1) težak, da se ne prevrne, in 2) lahek, da ga lažje izpraznimo. Sklepanje poteka na tak način naprej. Najti pravi kompromis je umetnost, za katero ni preprostega recepta. Nujno moramo doseči kompromis med kreativnostjo in zahtevami, ki so prej kot ne konzervativne.

Kompromise sklepamo tudi pri računalniškem programiranju. Pogosto se odločamo med velikostjo in hitrostjo programa oziroma algoritma. Tako npr. za digitalni video tipično velja, da bolj stisnjen video zahteva hitrejši računalnik za predvajanje zaradi bolj zapletenega postopka.

Kljub temu pa oblikovanje učinkovitih uporabniških vmesnikov v resnici ne zahteva kakih izrednih umetniških talentov. Še več, izkaže se, da pretirana inovativnost na tem področju prej zmede uporabnika kot pa pomaga. Ponavadi si lahko precej pomagamo z raznimi *modeli* in z nekaj preprostimi racionalnimi pravili. Osnovni koncepti uporabniških

vmesnikov so podani že v drugem poglavju. Tukaj bomo skušali podati nekaj dodatnih praktičnih pravil, ki lahko pomagajo programerju.

Sistem Unix na prvi pogled krši mnoga ustaljena pravila dobrega oblikovanja uporabniške izkušnje; šala pravi, da povprečen uporabnik Unixa nikoli ne ve, kako se ukaz "print" imenuje ta teden. Vendar je tak vtis zgolj površen. Unix je mnogo starejši od grafičnih uporabniških vmesnikov in v resnici zasnovan na nekaterih odličnih oblikovalskih principih, zaradi katerih po več kot tridesetih letih uporabe zgleda bolj živ in perspektiven kot kdajkoli prej. Ti principi, skupaj z mnogimi primeri iz prakse, so natačno obravnavani v odlični in na spletu prosto dostopni knjigi Erica S. Raymonda *The Art of Unix Programming* [3].

Filozofijo Unix povzemajo naslednja pravila:

1. Modularnost: Pišite enostavne dele povezane z jasnimi vmesniki.
2. Jasnost: Jasnost je boljša kot domiselnost.
3. Kompozicija: Oblikujte programe tako, da bodo povezani z drugimi programi.
4. Separacija: Ločite politiko od mehanizma; ločite vmesnik od motorja.
5. Enostavnost: Oblikujte za enostavnost; kompleksnost dodajte samo tam, kjer je ta nujna.
6. Minimalnost: Program naj bo velik le, če lahko demonstrirate, da ne bo šlo drugače.
7. Transparentnost: Oblikujte za transparentnost, da bosta nazdor in razhroščevanje lažja.
8. Robustnost: Robustnost je otrok transparentnosti in enostavnosti.
9. Reprezentacija: Shranite znanje v podatke, da bo program lahko neumen in robusten.
10. Najmanjše presenečenje: Pri oblikovanju vmesnikov vedno naredite najmanj presentljivo stvar.
11. Tišina: Če program nima nič presenetljivega za povedati, naj bo tiho.
12. Popravljanje: Če je nujno, da se naloga prekine, potem naj se to zgodi čimprej in opazno.
13. Ekonomičnost: Programerjev čas je dragocen, varčujte z njim bolj kot z računalnikovim časom.

14. Generiranje: Izogibajte se ročnemu delu; pišite programe ki pišejo programe, kadar lahko.
15. Optimiziranje: Naredite prototip pred končno verzijo. Poskrbite, da najprej dela, šele potem optimizirajte.
16. Različnost: Ne verjemite trditvam o eni pravi-in-edini metodi.
17. Razširljivost: Oblikujte za prihodnost ker bo tu prej kot si mislite.

3.4 Nadzor nad okoljem osrečuje uporabnika

Uporabnik je zadovoljen takrat, ko *uspešno* opravi neko *nalogo*, recimo natisne članek s spleta na tiskalnik. Taka naloga je navadno sestavljena iz podnalog ali korakov: 1) odpri spletni brskalnik (Mozilla) 2) poišči iskani članek v iskalniku, 3) prenesi članek na lokalni računalnik, 4) odpri članek v primernem pregledovalniku in 5) natisni na primernem tiskalniku. Vsakič, ko uporabnik katerega od korakov ne more uspešno opraviti, ima *problem*. Ta je lahko zelo preprost in *rešljiv* – npr. tiskalnik ni prižgan – ali pa zelo kompliciran in uporabniku *nerešljiv* – npr. sploh ne obstaja ustrezen gonilnik za naš tiskalnik.

Vendar kompleksnost problema uporabnika navadno ne zanima. Uporabnika pravzaprav ne zanima ničesar, kar ni neposredno povezano z nalogo. Če je nek problem za uporabnika nerešljiv, je iz njegovega zornega kota to praktično videti tako, kot če računalnik ne dela!

V zgornjem scenariju tiskanja članka je kako posebno znanje, neposredno povezano z nalogo bržkone smiselno samo pri uporabi iskalnika. Vse ostalo bi moralo biti očitno.

Težava so tudi rešljivi problemi. Majhni neuspehi pri opravljanju naloge – takrat, ko nekaj ni delovalo po pričakovanju in je moral zato uporabnik rešiti problem – se nabirajo v veliko nezadovoljstvo. V zvezi s tem je bilo opravljenih precej psiholoških raziskav; s tem se je ukvarjal Dr. Martin E. P. Seligman, ki je razvil teorijo *naučene nemoči* (angl. Learned Helplessness). Teorija trdi, da velik del depresij izvira iz občutka *nemoči*: občutka, da ne morete nadzorovati svojega okolja.

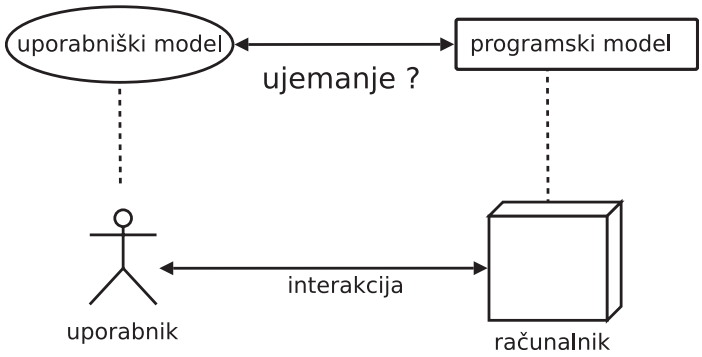
Bolj ko se počutite, da nadzorujete svoje okolje in da stvari dejansko *delujejo*, *srečnejši* ste. Če ste zlovoljni, je to bržkone zato, ker se je zgodilo nekaj, česar niste mogli nadzorovati, pa čeprav nekaj majhnega. Take frustracije so lahko majhne, vendar se seštevajo!

Definicijo dobrega uporabniškega vmesnika nam povzame naslednji kardinalni aksiom [5]:

Aksiom 1 *Uporabniški vmesnik je dober takrat, ko se program obnaša natanko tako, kot to pričakuje uporabnik.*

Vse ostalo so le posledice. Program mora uporabniku pomagati, da se osredotoči na nalogo. Ta izrek je tudi razlog, zakaj je pretirana nepremišljena "ustvarjalnost" pri oblikovanju uporabniških vmesnikov slaba. Nepričakovano obnašanje in presenečenja uporabnika ne razveselijo.

Uporabnikova predstava o tem, kaj se v programu dogaja, se imenuje *uporabniški model*. To je uporabnikova mentalna predstava o tem, kaj računalnik pravzaprav počne. Različni uporabniki imajo seveda različne predstave o tem, kaj se dogaja. Uporabniški model nekega specifičnega uporabnika izhaja iz 1) naučenega znanja uporabnika (programi so v meniju "Start"; gumb "Prekliči" ne bo ničesar zbrisal) ter 2) intuicije in zmožnosti posploševanja (kakšna je ikona za spletni brskalnik? *Katerikoli* spletni brskalnik?).



Slika 3.1: Ali je uporabniški vmesnik dober?

Tudi računalnik ima "mentalno predstavo" o tem, kaj se dogaja. Ta je zakodirana v bitih, ki jih obdeluje procesor ter se imenuje *programski model*, ki dejansko velja. Če programski model dobro ustreza uporabniškemu modelu, potem imamo uspešen uporabniški vmesnik.

Če se oba modela ne ujemata, je uporabnik zmeden, saj odziv računalnika ne ustreza njegovim pričakovanjem. Vsakič, ko neka uporabnikova akcija ne privede do pričakovanega rezultata, doživi uporabnik majhno frustracijo. Zelo pomembno je, da je vizualni odziv računalnika ustrezen, saj uporabnik svoj mentalni model naslanja na to, kar *vidi*.

Nekaj primerov:

1. Privzeto enojno klikanje na ikone na namizju KDE. KDE je menda idejo pobral iz brskalnika za splet, kjer je v navadi enojno klikanje. Uporabniki praktično katerega koli drugega namizja, vključujoč Macintosh, Windows in druga namizja Unix je to več kot 90 %, so navajeni odpiranja ikon z dvojnim klikanjem. Na privzetem namizju KDE ta akcija odpre dve enaki okni, pri čemer mora uporabnik 1) čakati dlje, ker se program sočasno nalaga dvakrat, in 2) zapreti

odvečno okno, s čimer se sicer ni imel namena ukvarjati. Uporabnik ni srečen, uporabniška izkušnja je slaba.

To je primer *inercije uporabniškega modela*. Uporabniki, ki prej niso uporabljali namizja Windows, imajo drugačen uporabniški model in se enojnega klikanja navadijo brez težav. Vendar je takih uporabnikov malo in prednosti enojnega klikanja niso bržkone dovolj prepričljive.

2. Ni odziva! Ko uporabnik izda nek ukaz v kratkem času, največ par sekund, pričakuje vizualni odziv programa. Vse kar traja dlje, znatno poslabša izkušnjo in produktivnost. Uporabnik *misli*, da ukaz ni deloval, in poskusi ponovno.

   Namizja Unix imajo zloglasno slab *odzivni učinek*. Ko na sistemu Unix poženemo nek program do trenutka, ko se odpre prvo okno, lahko mine več kot 10 sekund. V tem času ni nobenega odziva, da se je program pognal. Pojav je še nekoliko bolj izrazit na namizjih, ki se izvajajo s CD-ja, saj je takrat nalaganje še nekoliko počasnejše.

   Pri odzivnem efektu si lahko pomagamo s 1) spreminjanjem miškinega kazalca v uro in 2) pokazatelji dejavnosti. Primer slednjih najdemo v spletnih brskalnikih, npr. globus, ki se vrti med nalaganjem strani.

3. Očitni tiskalniki, na katere uporabnik lahko tiska, so: 1) lokalno priključeni tiskalniki, 2) tiskalniki, ki na omrežju oglašujejo tiskalniške storitve. Na voljo so vsi podatki za začetek tiskanja, kljub temu pa je uporabnik primoran odpreti kontrolno ploščo in s čarovnikom ročno dodati tiskalnik.

4. Urejevalnik besedila MS Word privzeto vklopljeno funkcijo za samodejno popravljanje besedila. Če v angleški Word pišete slovensko besedilo – kar sploh ni redko, saj večina uporabnikov privzetega jezika za besedilo ne spremeni v slovenščino – se beseda "teh" vztrajno spreminja v "the". Uporabnik to navadno opazi šele, ko je nekaj besedila že napisanega. Ko pride k sebi od neljubega presenečenja, mora 1) izklopiti samodejno popravljanje, česar najbrž ne zna, in 2) pregledati in popraviti do tedaj napisano. Veselje je neizmerno.

Kako vemo, kakšen je uporabniški model?

Najenostavneje je, da vprašamo kar uporabnike same. Veliko uporabnikov aplikacij računalnike načeloma pozna bolj površno in si stvari predstavlja na precej preprost način. Uporabnika prosimo, da nam čim bolj natančno razloži, kaj se po njegovem dogaja in kako si sam predstavlja uporabo.

Če ne najdemo primernih testnih uporabnikov, si jih lahko izmislimo. Na primer:

1. Nataša je profesorica angleščine in prevajalka. Računalnik kot izboljšan pisalni stroj uporablja že od leta 1980. Edina programa, ki ju pozna sta Microsoft Word in Wordstar, starinski urejevalnik besedil. Vse datoteke shranjuje v eno mapo. Povsem očitno je, da ne bo nikoli uporabljala ukazne lupine.

2. Janez uporablja Advanced Windows Server. Poleg tega ima nameščen še program za odstranjevanje virusov ter MS Office. Naredil je nekaj makro ukazov za Word.

3. Boštjan je slikar, ki se za dodaten zaslužek ukvarja z ilustracijo. Pozna programe Photoshop, Illustrator in Premiere podjetja Adobe. Ponavadi se mu mudi z roki. Izdelki morajo biti ustrezne kakovosti (npr. barve v formatu CMYK, če gre za tiskanje).

Tudi taki izmišljeni preprosti opisi nam lahko dokaj dobro pričarajo vtis, s kakšnimi uporabniki imamo opravka.

Programski model ima lahko več nivojev, pri čemer navadno gledamo najvišjega, ker je najenostavnejši – temu smo na začetku poglavja rekli *konceptualni nivo*. Tudi uporabniki vedno najraje izberejo najenostavnješi model, ki ustreza njihovemu znanju in potrebam.

Izrek 1 *Če programski model ni trivialen, potem to verjetno ni uporabniški model.*

Zanimiv primer so ravno urejevalniki besedil. Njihov najpogostejši uporabniški model je "pisalni stroj s televizorjem" – ravno zaradi enostavnosti. Urejevalniki, kot so OpenOffice.org in MS Word ta model odlično podpirajo, zlasti zaradi WYSIWYG in neposrednosti. Vendar ta model odlično deluje predvsem za kratka besedila. Stavljanje bolj zapletenih besedil in knjig zahteva kompleksnejši model, kakršnega podpira denimo LaTeX (OpenOffice.org Writer in MS Word ga sicer tudi podpira, vendar uporabniku ni takoj očiten). Zanimivo je to, da je LaTeXov model za stavljenje kompleksnih besedil pravzaprav enostavnejši od tistega, ki ga podpira Word, saj vsebuje vse nujne elemente za stavljenje znanstvenih knjig. Precej uporabnikov Worda pa ne zna naprimer niti številčiti slik, kar nam LaTeX pri privzeti predlogi naredi avtomatsko. Več o različnih pristopih k oblikovanju besedil je v uvodu k 5. poglavju.

Realizacija na videz preprostega programskega modela lahko zahteva izredno zapleteno podporno infrastrukturo. Urejevalnik Emacs je namenjen na videz preprosti nalogi urejanja besedila, vendar je precej zakompliciran.

Vabe

Pred nekaj leti se je močno razširila uporaba gumbov, ki imajo 3D videz; gumb je videti tako, kakor da izstopa iz ekrana.

To je primer uspešne *vabe* (angl. *affordance*). Taki gumbi namreč kar kličejo po tem, da kliknete nanje. Drug primer uspešne vabe so zavihki (tabi) – uporabnik intuitivno ve, kako delujejo.

Problem z neprimernimi vabami je pogosto prisoten na spletnih straneh – te so pogosto oblikovanje tako, da ni razlike med napisom, na katerega lahko kliknete in napisom, ki je samo napis. To zanesljivo zmede uporabnika. Napisa vendar *zgledata* enako! Kako to, da lahko na enega kliknete, na drugega pa ne?

3.5 Ponujanje pretirane možnosti izbire je slabo

Skoraj vsak program ima okno, v katerem lahko uporabnik določi vrednosti raznim nastavitvam – ponavadi ga najdemo pod ukazom nastavitve ali "preferences". To okno je v resnici zgodovinski zapis razvoja programa – prikazuje odločitve, pri katerih se avtorji programa niso mogli odločiti, kaj je prav. Uporabniku je navadno vseeno, v kakšnem privzetem zapisu program shranjuje datoteke – pomembno je, da stvar deluje!

Izrek 2 *Vsakič, ko program ponudi izbiro med več možnostmi, od uporabnika zahteva, da sprejme odločitev.*

V resnici izbire same po sebi niso problematične. Svoboda izbire je čudovita stvar. Problem nastane pri izbirah, ki uporabnika *ne zanimajo*. To pa je vse, kar ni povezano z nalogo. Vsakič, ko od uporabnika zahtevamo odločitev v zvezi z nečim, kar ni povezano z nalogo prekinemo njegov delovni proces, kar je izredno moteče. Naloga oblikovalca uporabniškega vmesnika je, da sam naredi vse te nezanimive odločitve namesto uporabnika.

Primer: ko na nekaterih starejših inačicah sistema MS Windows odprete pomoč, se pokaže okence, ki vpraša, ali naj zgradi indeks pomoči tako, da:

1. minimizira velikost podatkovne baze,
2. maksimizira možnosti iskanja ali
3. možnosti po meri.

Ko uporabnik odpre pomoč, navadno že tako ni preveč zadovoljen, saj stvari niso delovale po pričakovanju – sicer najbrž ne bi iskal pomoči.

Zdaj pa ga to okence zmoti *že drugič* in ga sprašuje nekaj, česar najverjetneje sploh ne razume in je popolnoma nepovezano z nalogo. Tako okence bo uporabnika prej ali slej zagotovo spravilo v slabo voljo.

3.6 Spoštovanje do uporabnika

Pomembna kategorija oblikovanja uporabniških vmesnikov je spoštovanje do uporabnika. Uporabnika spoštujemo tako, da ga ne spoštujemo. To načelo ponazorimo z naslednjimi izreki:

Izrek 3 *Uporabniki ne berejo navodil*

Uporabniki navadno posežejo po navodilih šele takrat, ko se kaj zaplete, to pa je nekaj, čemur se skušamo na vso moč izogniti. Za programe naj sploh ne bi potrebovali navodil, kakor jih ne potrebujemo za televizor ali pralni stroj.

Ena od lastnosti modernih grafičnih vmesnikov naj bi bila, da programi vsi delujejo na podoben način; vsi imajo menije "Datoteka" in "Urejanje" na začetku, kombinacija tipk `<Control>+<C>` kopira izbrani objekt, itd. Znanje, pridobljeno pri uporabi enega programa, naj bi se v čim večji meri preneslo tudi na druge programe, ki delujejo v istem okolju.

Navodila so smiselna, če vsebujejo dodatne ali koristne informacije v zvezi z domeno, na katero se nanaša program – npr. navodila za Photoshop vsebujejo kratek tečaj priprave materialov za tisk. Pogost razlog, da uporabniki ne berejo navodil je tudi to, da so velikokrat slaba.

Uporabniki pravzaprav ne berejo ničesar!

Uporabniki pogosto ne preberejo okenc, ki na dolgo in široko razlagajo določeno zadevo, temveč kar kliknejo na najbolj očitno izbiro. Predstavljajte si neprijetno presenečenje, ki bi nastalo, če bi ob kliku na gumb "Prekliči" program izbrisal podatke!

Izrek 4 *Uporabniki ne zmorejo dobro nadzorovati miške*

Uporaba miške zahteva določene motorične spretnosti, ki so lahko pri posameznikih slabše, denimo zaradi neizkušenosti ali invalidnosti. Tudi nasploh velja, da je do piksla natančno klikanje precej naporno.

Primeri:

1. V urejevalnikih besedila MS Word in OpenOffice.org imamo v funkcijski vrstici meni s slogi za logično oblikovanje besedila, zraven

pa gumbe za trdo oblikovanje. Kljub očitnim prednostim logičnih slogov velika večina začetnikov uporablja trdo oblikovanje. Zakaj?

Odgovor na to vprašanje se skriva v slabem meniju s slogi. Čeprav je prostora na zaslonu ogromno, je padajoči meni zelo kratek – uporabnik mora z mučnim klikanjem in premikanjem drsnika na majhnem mestu poiskati želeni slog, in to vsakič, ko želi spremeniti slog. Neprimerno manj naporen je klik na gumb "Krepko", ki ga ni treba iskati.

2. Uporaba proporcionalnih pisav v okencih za vpis besedila je slaba, čeprav izgleda lepše in na videz bolj skladno z ostalimi deli grafičnega vmesnika. Med črkama l in i v besedi lilije je v običajni proporcionalni pisavi na zaslonu prostora okoli 2 piksla in še tega prekrije miškin kazalec. Zadeti pravo mesto je izredno težavno. Primerjajte z `lilije`.

Izrek 5 *Uporabniki si stvari ne zapomnijo*

Lep primer tega so uporabniki, ki si razne postopke pri uporabi računalnika brez pretiranega razmišljanja pišejo na razne listke, kar ni posebej intuitivno, vendar pogosto deluje. Odlično orodje za rešitev tega problema so *čarovniki*, okenca ki nalogo razdelijo na korake. Pri tem ob vsakem koraku uporabnik vnese samo bistvene informacije. Čarovniki so pravzaprav zelo učinkovita izvedba menijskega vmesnika.

3.7 Primer: Slix

Zanimivo dejstvo je, da živi Linux (glej poglavje 4.2) dokaj dobro upošteva navedena pravila – avtomatsko razpoznavanje priključene strojne opreme je stvar, ki so jo veseli vsi uporabniki.

Slix je primer živega Linuxa, pri katerem so avtorji skušali upoštevati navedena pravila v čim večji meri in uporabniško izkušnjo še dodatno izboljšati. Pri tem so imeli naslednje konkretne želje in smernice:

1. Vse bistvene naloge morajo biti izvršljive preko grafičnega vmesnika. Uporabniku naj ne bi bilo nikoli potrebno uporabiti ukazne lupine, če tega noče.

2. Ukazna lupina mora biti enostavno dostopna za bolj izkušene uporabnike. Kot zelo uspešen se je pokazal gumb, ki upravljalnik datotek Konqueror razdeli na pol, pri čemer se v spodnji polovici pojavi lupina v tistem imeniku, ki ga upravljalnik datotek kaže. Navigacija do določenega imenika je tako neposredna in hitra.

3. Stvari morajo delati čim bolje s čim manj nastavljanja. Poudarek mora biti na nalogah.

4. Izdelek mora biti vizualno kakovosten. Uporaba barv mora biti zmerna. To je tisto, čemur Edgar Tufte pravi "veliko informacije" in "malo črnila" [5, 6, 7]. Ikone morajo dobro in očitno predstavljati metafore, ne pa neke vizualne mode.

5. Veliko malih izboljšav pomaga.

Slix smo preizkušali s petimi uporabniki različnih profilov. Praktično vsi problemi, ki smo jih opazili, so bili posledica nepričakovanega obnašanja. Naredili smo vrsto malih sprememb, ki so se izkazale kot dokaj uspešne in dobrodošle:

1. Ikone za "Moj računalnik" in "Omrežje" na namizju. Uporabniki sistema Windows so brez njih izgubljeni.

2. Popravek programa LinNeighborhood za pregledovanje lokalnega omrežja. Dodali smo funkcijo, ki na dvoklik odpre izbrano mapo v upravljalniku datotek. Prej je bilo potrebno uporabiti meni, dvoklik pa je bil brez funkcije. Odstranili smo vse nepomembne gumbe iz orodjarne.

3. Dodali smo pokazatelj zasedenosti računalnika v obliki ognja. Izkaže se, da je to zelo dobra metafora.

3.8 Primer: pisarniški programski paket OpenOffice.org

OpenOffice.org je paket pisarniških programov, ki v celoti ponuja rešitve za pripravo dokumentov, grafičnih in podatkovnih vsebin. Paket OpenOffice.org je nastal iz funkcionalno in oblikovno zelo sorodnega paketa StarOffice podjetja Sun. Slednji je v svojem začetku predstavljal brezplačno alternativo integriranih pisarniških orodij, kot so Microsoft Office, Lotus SmartSuite, Corel WordPerfect Office in AppleWorks. OpenOffice.org je nastal, ko se je podjetje Sun odločilo svoj izdelek tržiti.

Prednost pisarniških zbirk je v združljivosti posameznih modulov, kar omogoča hitro in enostavno izmenjavo podatkov med dokumenti. Delovno okolje z enotnim uporabniškim vmesnikom naj bi omogočalo čim bolj pregledne prenose med urejevalnikom besedil, preglednico ali predstavitvijo.

StarDivision, podjetje, ki je začelo z razvojem zbirke, je od leta 1999 v lasti podjetja Sun. Takoj po prevzemu podjetja je Sun napovedal tudi

brezplačno distribucijo pete generacije zbirke (StarOffice 5.1). Zbirka je dostopna na več platformah, kot so operacijski sistemi Windows 95/98/NT, Linux, Solaris Intel, SPARC in OS/2. OpenOffice.org se ponaša z večino lastnosti preshodnika in vsebuje podmnožico modulov iz paketa StarOffice. OpenOffice.org vsebuje naslednje module:

OpenOffice.org Writer – močan urejevalnik besedil, ki med drugim omogoča tudi preverjanje črkovanja, popoln nadzor nad slogi, upravljanje z različicami ter ustvarjanje spletnih strani (dokumentov HTML).

OpenOffice.org Calc – omogoča delo s preglednicami, v katere lahko vnašamo numerične podatke, formule ali besedilo ter z njimi operiramo, jih ponazarjamo z grafikoni ali izvozimo v druge module.

OpenOffice.org Impress – modul za izdelavo predstavitev, ki jih uporabljamo kot medij za posredovanje informacij občinstvu. Podatke, besedilo in grafikone, ki jih bomo predstavili, lahko tudi uvozimo iz drugih modulov.

StarOffice.org Draw – modul za vektorsko grafiko. Omogoča ustvarjanje vektorskih slik, bodisi kot samostojnih izdelkov ali pa kot del drugih dokumentov.

OpenOffice.org Math – matematični modul, ki omogoča urejanje in izračunavanje matematičnih dokumentov.

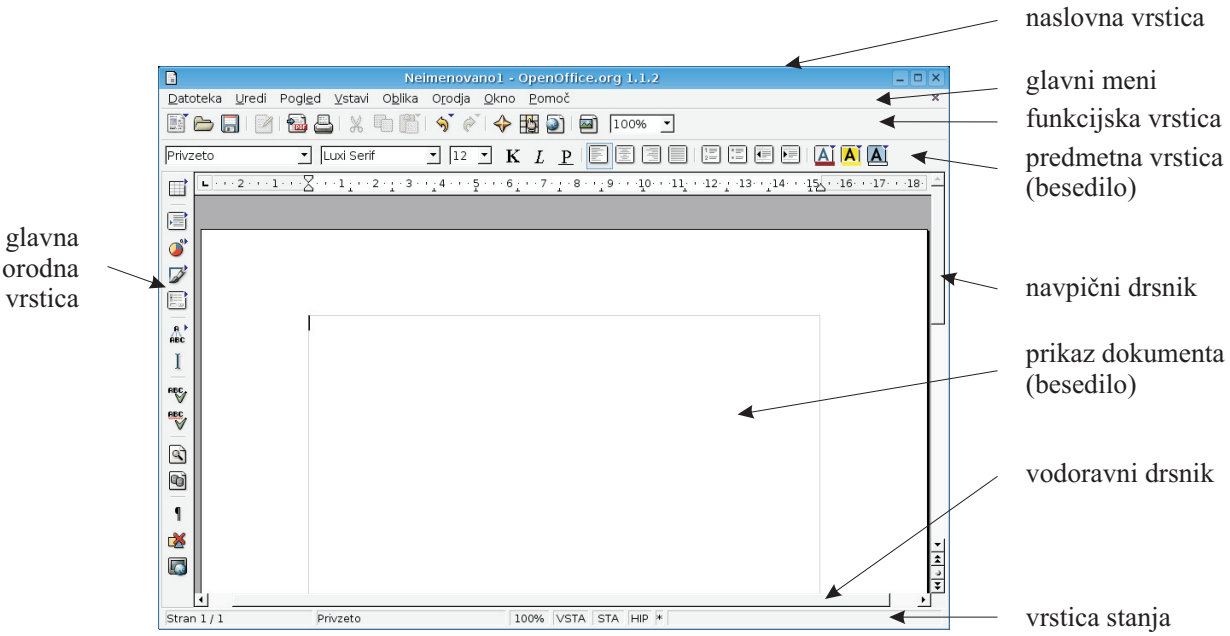
OpenOffice.org Chart – okolje za izdelavo grafov.

OpenOffice.org Basic – vgrajen skiptni jezik.

V nadaljevanju bomo opisali uporabniški vmesnik programov paketa OpenOffice.org in podali osnovne napotke za učinkovito delo. V prvi izdaji te knjige smo opisali delo s paketom StarOffice različica 5.1 za operacijski sistem Linux, v drugi izdaji, v drugi izdaji pa različico 5.2. Slednja je bila zadnja različica, ki jo je bilo moč brezplačno prenesti iz Sunovih spletnih strežnikov. V tretji izdaji zato obravnavamo paket OpenOffice.org, ki ga lahko uporabljamo brezplačno.

3.8.1 Uporabniški vmesnik

Programi zbirke OpenOffice.org imajo enoten videz in uporabniški vmesnik. Elementi vmesnika se dinamično prilagajajo tako vrsti dokumenta kot izbranim elementom v dokumentu. Tako se lahko uporabnik privadi dela z enim programom, podobna opravila v ostalih programih pa bo hitro spoznal in jih uporabil. Hkrati pa programi



Slika 3.2: Skupni elementi grafičnega vmesnika OpenOffice.org na primeru praznega dokumenta OpenOffice.org Writer

omogočajo udobno delo in enostaven dostop do najpogostejših lastnosti in ukazov elementov, ki jih trenutno ureja. Za nameček pa lahko vsak uporabnik okolje prilagodi svojim potrebam in opravilom. Videz programa iz zbirke OpenOffice.org ter njihovih elementov prikazuje slika 3.2.

Meniji

Na samem vrhu okna vsakega od programov se nahaja standarden **glavni meni**, od koder je dosegljiva velika večina ukazov. Vsak meni združuje ukaze po kategoriji, katere ime prikazuje izbira v vrstici menija. Ime menija ima po eno podčrtano črko, ki v kombinaciji s tipko `<Alt>` predstavlja bližnjico do tega menija. Meni "Datoteka" tako prikličemo s kombinacijo `<Alt>+<D>`. Meni lahko prikličemo tudi pritiskom na tipko `<F10>`, sicer pa nanj kliknemo z miško.

Imena glavnih menijev so preprosto prepoznavna in se nanašajo na določene predmete ali akcije. Nekateri od menijev se pojavljajo v vseh programih, na primer "Datoteka" za upravljanje datotek, "Uredi" za urejanje dokumenta, "Pogled" za nadzor nad prikazom, "Orodja" za dostop do pomožnih orodij (konfiguratorji, slovarji itd). Večina programov ima menije "Vstavi" za izbiro elementov, ki jih lahko v dokument vstavimo, "Oblika" za določanje in spreminjanje dokumenta ali izbranih

elementov, “Okno” za upravljanje oken in “Pomoč” za dostop do pomoči.

Tudi izbire v posameznem meniju imajo s podčrtano črko označeno tipko, ki jo pritisnemo za hitrejši dostop do izbire. Med izbirami se sicer pomikamo s smernimi tipkami. Nekateri meniji imajo podmenije, takšne menije spoznamo po puščici na desni strani izbire. Nekatere možnosti so lahko vklopljene ali izklopljene, pri čemer v prvem primeru na levi strani dobijo kljukico. Na desni strani izbire imajo nekatere možnosti izpisano kombinacijo tipk, ki prikličejo možnost iz programa brez dostopa do menijev. V splošnem velja dogovor, da se imena možnosti, ki odprejo pogovorno okno, končajo s tremi pikami.

Za delo s programi v grafičnem načinu torej ne potrebujemo natančnega znanja o vsakem ukazu ter leksikografskega poznavanja bližnjic na tipkovnici do ukazov. Zadostuje, da uporabnik ve, kaj želi v danem trenutku narediti, formulirati želeno v naravnem jeziku ter poiskati ustrezno zaporedje menijev in podmenijev, ki vodijo do ukaza. Če želimo na primer v dokument z besedilom vstaviti sliko (grafiko), odpremo meni “Vstavi” in poiščemo možnost “Grafika”. Ta nam odpre podmeni, iz katerega izberemo “Iz datoteke...”. Tri pike za imenom izbire nam pove, da se bo odprlo pogovorno okno, v katerem lahko izberemo datoteko s sliko. V nadaljnjem besedilu bomo takšno pot skozi menije navajali krajše kot “Vstavi→Grafika→Iz datoteke...”.

Poleg glavnega in njemu podrejenih menijev poznamo še **priročni meni**. Tega prikličemo s klikom desne miškine tipke na izbran predmet v dokumentu. V tem meniju so strnjene najpogostejše možnosti glavnega menija, ki se nanašajo na izbrani predmet.

Orodne vrstice

Pogosto uporabljene ukaze ali izbire lahko najdemo na **orodnih vrsticah**, ki vsebujejo orodja – gumbe in padajoče menije. Teh vrstic je več, vsaka pa združuje ukaze določene vrste. Orodne vrstice lahko prestavljamo z vlečenjem miške, skrijemo ali prikažemo z desnim klikom na meni ali neko vrstico, oz. preko “Pogled→Orodne vrstice”. Orodja imajo obliko gumbov, ki prikličejo pogovorno okno, takšnih, ki ostanejo pritisnjeni, ob naslednjem kliku pa se vrnejo v nepritisnjeno stanje. Nekateri prikličejo padajoči meni, seznam ali drugi grafični element – te spoznamo po majhni modri navzdol obrnjeni puščici. Če na takšnem gumbu pritisnemo in držimo levo tipko miške, bomo priklicali padajoč meni, iz katerega nato izberemo želeno vrednost. Gumb bo nato prevzel vrednost zadnje izbire, in če kratko kliknemo nanj, ponovimo zadnjo izbiro.

Večina gumbov ima majhno sliko, ikono, ki ponazarja pomen pripadajočega gumba. Večinoma je ikona dovolj jasna, če pa nismo prepričani v njen pomen, pa si lahko pomagamo z “nasvetom”. “Nasvet”

je besedilo v oknu, ki se pojavi, če miškin kazalec postavimo nad ikono in ga za nekaj sekund ne premikamo.

Funkcijska vrstica je v privzetem načinu takoj pod glavnim menijem. Vsebuje ukaze, ki se nanašajo na dokument in so skupni vsem programom: ustvarjanje novega dokumenta (padajoči meni, iz katerega lahko izberemo vrsto novega dokumenta), odpiranje datoteke, shranjevanje. Gumb "Uredi datoteko" omogoči urejanje dokumenta, ki je nastavljen samo za branje. Sledi ukaz za izvoz dokumenta v zapis PDF ter ukaz za tiskanje dokumenta. Naslednja skupina omogoča hitrejše urejanje besedila: ukazi "Izreži" in "Kopiraj" izrežeta oz. skopirata izbrane elemente dokumenta na odložišče, "Prilepi" pa vsebino odložišča vstavi v dokument. Ukaz "Razveljavi" je izredno koristen, saj omogoča razveljavljanje posameznih korakov, ki smo jih izvedli od odpiranja dokumenta. "Uveljavi" ponovi predhodno razveljavljene akcije. Gumba "Krmar" in "Slogovnik" prikažeta ali skrijeta ustrezno plavajoče okno, gumb "Hiperpovezave" pa prikliče pogovorno okno, v katerem nastavljamo povezavo, na katero bo kazal izbran predmet dokumenta.

Predmetna vrstica vsebuje najpogostejše ukaze, ki jih izvajamo nad izbranim elementom. Vsebina te vrstice se dinamično spreminja v odvisnosti od izbora elementov v dokumentu. Hkrati stanje gumbov in drugih elementov vrstice odraža lastnosti izbranega predmeta. Lahko se zgodi, da imamo možnost izbire med več kot eno predmetno vrstico (npr. če v dokumentu z besedilom v programu OpenOffice.org Writer vklopimo številčenje). Takrat med posameznimi vrsticami preklapljamo z gumbom na desni strani, ki prikazuje puščico v desno, ali s priročnim menijem vrstice.

Na levi strani ima vsak program svojo **glavno orodno vrstico**. Večina gumbov te vrstice ima majhno puščico, ki kaže v desno. Č tak gumb pritisnemo in držimo tipko miške, bomo priklicali pripadajočo orodno vrstico. Ta lahko ob izbiri orodja izgine, če pa povlečemo za njegovo naslovno vrstico, pa orodna vrstica ostane plavajoča nad dokumentom. Glavna orodna vrstica je izhodišče za orodne vrstice, ki so bližnjice za vstavljanje različnih elementov ali predmetov v dokument. Vsebuje tudi bližnjice do različnih funkcij, ki so lastni določeni vrsti dokumentov, zato ne sodijo na funkcijsko vrstico.

Vrstice lahko prilagodimo svojemu delu in pogostim operacijam. S klikom desne miškine tipke na orodno vrstico prikličemo priročni meni, v katerega zgornjem delu določamo, katere vrstice so vidne in katere skrite. V podmeniju "Vidni gumbi" določamo vsebino izbrane vrstice. S "Konfiguriraj..." in "Po meri..." lahko dodadno spreminjamo videz in vsebino vrstic. Možnost "Ponastavi" pa povrne vsebino vrstice na privzeto stanje.

Naslovna vrstica in vrstica stanja

Vsako okno ima v svojem zgornjem delu naslovno vrstico. Glavno okno programa v naslovni vrstici vsebuje ime dokumenta oz. pripadajoče datoteke. Pogovorna okna in plavajoče orodne vrstice vsebujejo v naslovni vrstici ime pripadajočega okna.

Vrstico stanja ima glavno okno programa na svojem dnu. Vrstice stanja različnih dokumentov se med seboj razlikujejo, vse pa prikazujejo pomembnejše podatke o trenutnem dokumentu, pogledu, načinu vnosa in mestu kazalke. Nekatera polja vrstice stanja se odzovejo na klik z levo tipko miške, z dvojnim klikom na druge pa lahko prikličemo pogovorno okno, kjer nastavljamo vrednost, ki jo vrstica stanja na tem mestu prikazuje.

Povleci in spusti

Grafična okolja stanje vsebine dokumentov prikazujejo grafično. S kazalcem miške lahko predmete dokumenta ali ikone v različnih programih prenašamo po grafičnem vmesniku in tako na intuitiven način izvedemo določene akcije.

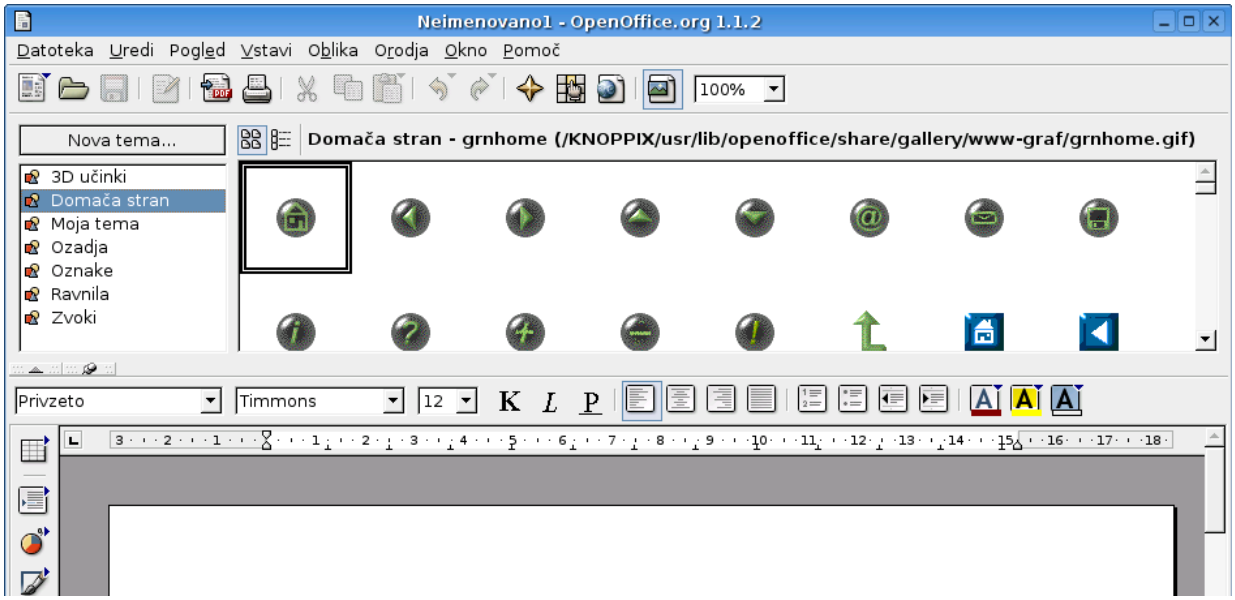
Akcijo povleci in spusti začnemo tako, da nad predmet ali ikono predmeta pripeljemo kazalec miške ter pritisnemo in zadržimo gumb miške. S premikanjem miške nato premikamo ponazoritev predmeta po zaslonu. Predmet lahko nesemo na drug del istega okna ali ga odnesemo na področje okna drugega programa. Ko tipko miške sprostimo, program izvede akcijo, ki pa je odvisna od od tipa predmeta ter od mesta prenosa. Tako lahko prestavljamo predmete dokumenta znotraj ali med dokumenti, prenašamo datoteke s prestavljanjem njihovih ikon med okni ipd.

Galerija

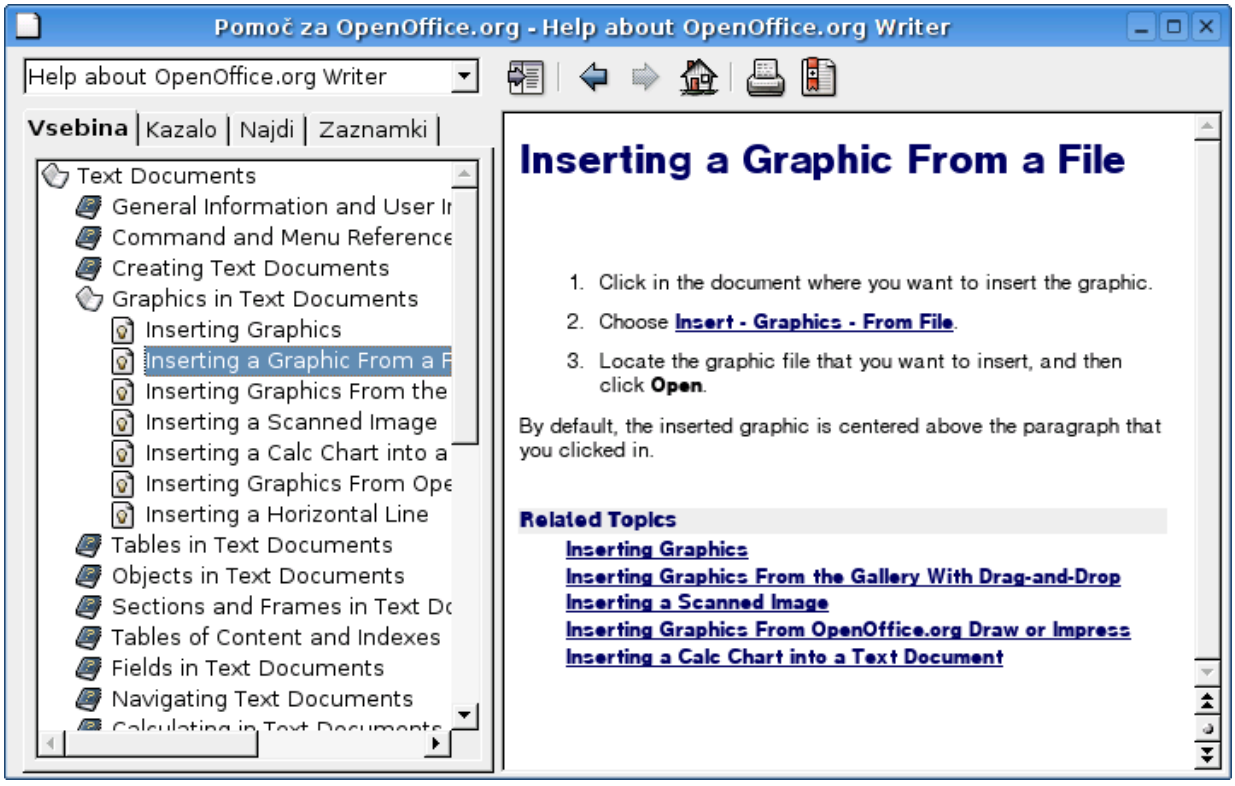
Galerija je posebna orodna vrstica, ki vsebuje zbirke slikovnih in drugih predmetov, shranjenih na disku (slika 3.3). Prikažemo in skrijemo jo s tipko  funkcijske vrstice oz. z “Orodja→Galerija”. Galerija se odpre v zgornjem delu okna.

Na levi strani okna z galerijo izbiramo med temami, ki združujejo posamezne predmete. Na desni strani vidimo prikazane predmete (slike, zvoki ipd.) izbrane teme. Predmete iz galerije lahko vstavimo v dokument tako, da jih povlečemo na območje dokumenta ali kliknemo z desno miškino tipko na predmet in določimo želeno akcijo iz priročnega menija.

Teme lahko dodajamo tudi svoje in urejamo seznam datotek, ki spadajo k temi.



Slika 3.3: Galerija



Slika 3.4: Primer okna s pomočjo

Sistem pomoči

Programi imajo običajno nameščeno zbirko elektronskih dokumentov, ki razložijo elemente programa, funkcije ter uporabo programa. Te dokumente imenujemo tudi pomoč (angl. *help*).

Sistem pomoči je vedno dosegljiv z izbiro "Pomoč→ Vsebina" v glavnem meniju. Pojavi se okno (slika 3.4), ki ima na levi strani podokno za krmarjenje. Podokno lahko skrijemo ali prikažemo z gumbom "Skrij/Prikaži podokno za krmarjenje". To podokno omogoči izbiro med vsebino, kazalom, iskanjem ter zaznamki. Na vrhu podokna iz padajočega menija izberemo program, za katerega iščemo pomoč. Na desni strani vidimo trenutno stran pomoči, nad katero je nekaj gumbov za navigacijo po pomoči. Vsaka stran pomoči je v obliki hiperbesedila – kopice dokumentov, ki so med seboj povezani v strukturo, po kateri se premikamo preko povezav – hiperpovezav.

Ob aktiviranju pomoči se prikaže uvodna stran (kazalo) za aktivni program (npr. OpenOffice.org Writer). Z izbiro povezave se premikamo proti temi, ki nas zanima. Pomagamo si z ikonami "Nazaj", "Naprej" in "Prva stran". Med navigacijo si lahko pomembnejše strani izpišemo na tiskalnik z "Natisni..." ali zaznamujemo z "Dodaj k zaznamkom"

Č kjerkoli v programu pritisnemo tipko `<F1>`, bomo priklicali stran pomoči, ki se navezuje na trenutno okolje.

Pomočnik je del sistema pomoči, ki se v obliki majhne slike pojavi v desnem spodnjem kotu okna vsakič, ko program zazna, da bo uporabnik potreboval pomoč pri izvajanju kakšne naloge. S klikom na sliko pomočnika prikličemo ustrezno temo pomoči. Pomočnika izklopimo ali omogočimo preko "Pomoč→Pomočnik".

Nasveti so imena gumbov in drugih orodij, ki se pojavijo, če nad posamezno orodje pripeljemo kazalec miške in ga tam pustimo za kratek čas. Nasvete izklopimo ali omogočimo preko "Pomoč→Nasveti".

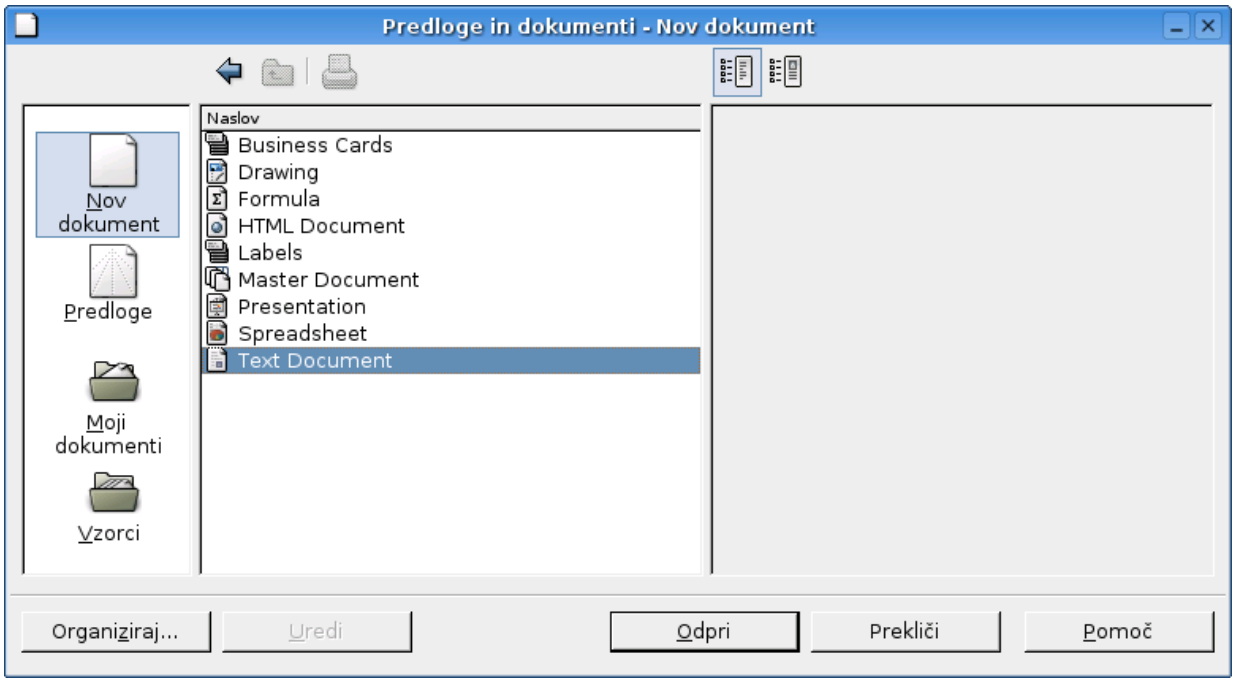
Razširjeni nasveti delujejo podobno kot nasveti, vendar namesto imena prikažejo krajšo razlago orodja, nad katerim za krajši čas pustimo kazalec miške. Razširjene nasvete izklopimo ali omogočimo preko "Pomoč→Razširjeni nasveti". S kombinacijo `<Shift>+<F1>` prikličemo kazalec pomoči, s katerim se sprehajamo po orodjih, program pa nam sproti prikazuje razširjene nasvete.

3.8.2 Delo z dokumenti in datotekami

Dokumente hranimo v obliki datotek in tako omogočimo kasnejše branje, urejanje ter elektronsko izmenjavo naših izdelkov. Datoteka hrani tako vsebino kot obliko in dodatne elemente v natanko taki obliki, kot ga je

imel dokument v trenutku, ko smo ga shranili. Vrsta datoteke odraža vrsta dokumenta, zapis, ki ga zna prebrati pripadajoči program (in nekateri sorodni programi), za lažjo prepoznavo pa ima ime datoteke tričrkovno končnico, ki označuje vrsto zapisa.

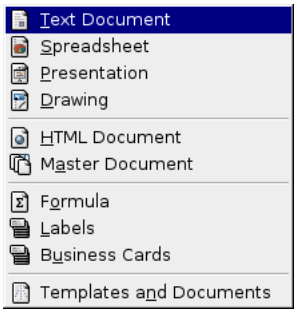
Nov dokument



Slika 3.5: Pogovorno okno za ustvarjanje novega dokumenta

Prazen dokument ustvarimo tako, da poženemo program, s katerim bomo dokument urejali. V namizju KDE naredimo to tako, da kliknemo na tipko "K" in iz menija izberemo možnost "Pisarna". V podmeniju izbiramo med programi iz zbirke OpenOffice.org ter nekaterimi predlogami, ki uporabljajo te programe. Izberemo lahko tudi "OpenOffice.org iz predloge", ki prikliče pogovorno okno za ustvarjanje novega dokumenta, dokumenta iz predloge ali odpiranje obstoječega dokumenta (slika 3.5).

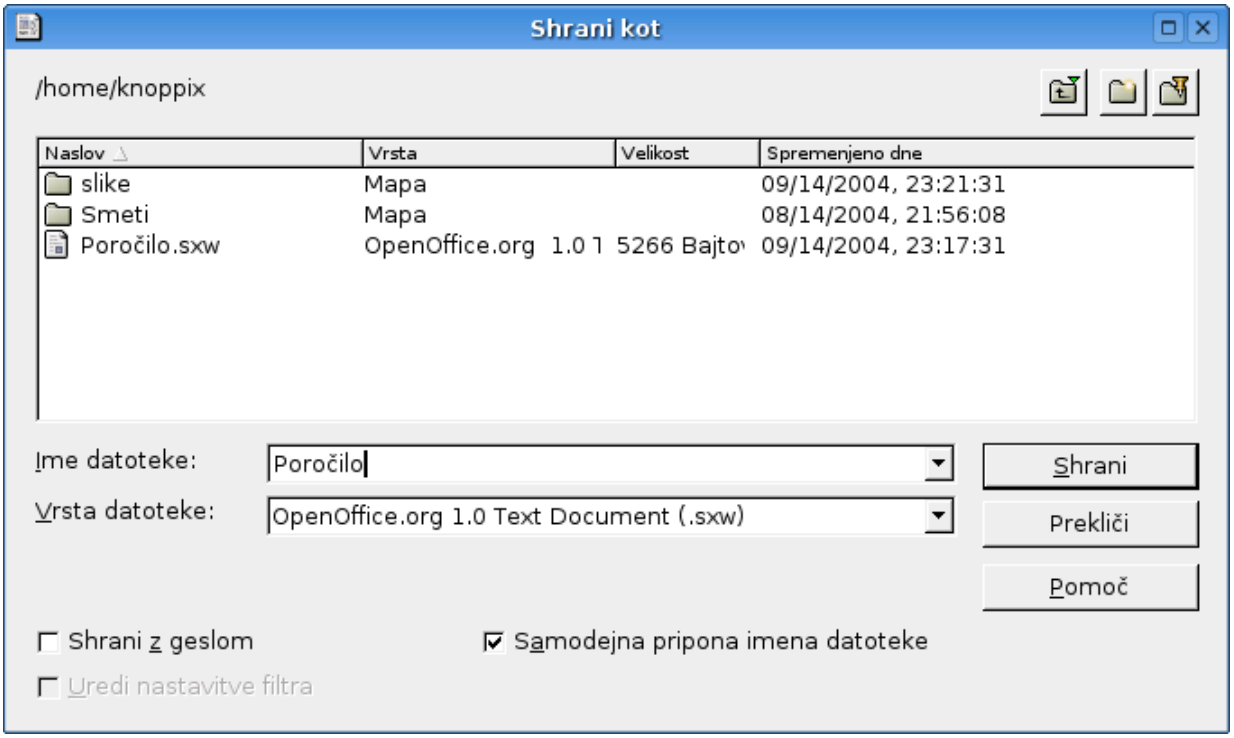
Ko imamo odprt že vsaj en program iz zbirke OpenOffice.org, lahko ustvarimo nov dokument s pomočjo "Datoteka→Nova→...", ki nam odpre podmeni s slike 3.6. Izbiramo lahko med dokumentom z besedilom (*Text Document*), preglednico (*Spreadsheet*), govorno predstavitvijo (*Presentation*), risbo (*Drawing*), spletnim dokumentom (*HTML Document*), glavnim dokumentom, ki združuje več dokumentov (*Master Document*), matematičnim obrazcem (*Formula*), nalepkam (*Labels*),



Slika 3.6: Podmeni z izbirami za ustvarjanje novega dokumenta

vizitkam (*Business Cards*) ter pogovornim oknom za ustvarjanje novega dokumenta s slike 3.5 (*Templates and Documents*).

Shranjevanje dokumentov



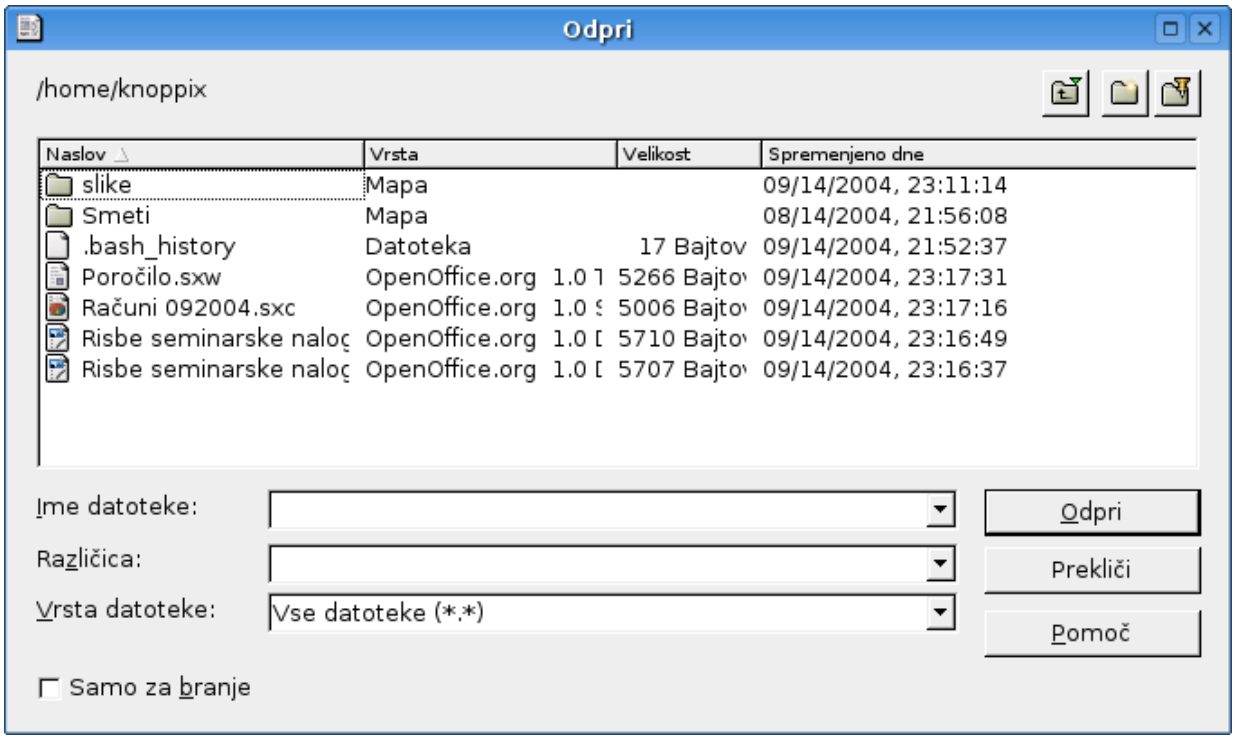
Slika 3.7: Pogovorno okno za shranjevanje dokumenta

Novoustvarjen dokument ima generično ime, pred izgubo izdelka pa se kasneje zavarujemo s shranjevanjem. Dokument, ki še nima imena datoteke oz. bi ga radi shranili v drugo datoteko, shranimo z "Datoteka→Shrani kot...". Spremembe, ki smo jih v dokument naredili od

zadnjega shranjevanja, pa shranimo v isto datoteko z “Datoteka→Shrani”.

“Dokument→Shrani kot...” prikliče pogovorno okno za shranjevanje, kot ga vidimo na sliki 3.7. V zgornjem levem delu okna vidimo pot do mape, katere vsebino vidimo v osrednjem delu okna. Prikaz je običajno omejen na podrejene mape in na datoteke s končnico, določeno v rubriki “Vrsta datoteke”. S klikom na gumb se pomaknemo v nadrejeno mapo, če pa gumb držimo, se nam odpre padajoči meni z nekaj več možnosti. V podrejene mape se pomikamo z dvojnim klikom na ime mape v osrednjem delu okna. Č se med navigacijo izgubimo ali se želimo hitro vrniti v svojo domačo mapo, kliknemo . S klikom na gumb ustvarimo novo mapo. Ko smo zadovoljni z mestom za shranjevanje dokumenta, v “Ime datoteke” vnesemo ime, ki jo bo nosila datoteka, in kliknemo “Shrani”.

Odpiranje dokumentov



Slika 3.8: Pogovorno okno za odpiranje dokumenta

Dokument v programu odpremo z “Dokument→Odpri...”. Pogovorno okno, ki se odpre, vidimo na sliki 3.8 in je zelo podobno oknu za shranjevanje.

Ko v osrednjem delu pogovornega okna kliknemo dokument, ki ga želimo odpreti, kliknemo “Odpri”.

Podobno pogovorno okno se odpre vsakič, ko želimo vstaviti ali odpreti dokument, shranjen v datoteki, npr. sliko.

Predmeti drugih programov v dokumentu

Programi iz zbirke OpenOffice.org podpirajo sodelovanje med dokumenti preko protokola OLE (*Object Linking and Embedding* – gnezdenje in povezava predmetov), sicer znanega iz okolja Windows. S tem protokolom lahko znotraj posameznega dokumenta ustvarimo predmet, ki je dokument drugega programa. Predmet nato urejamo z vmesnikom programa, ki mu dokument pripada, čeprav je vgnezden v drug dokument. Vgnezdene dokumente potem shranimo skupaj z nosilnim dokumentom.

Predmet OLE vstavimo z "Vstavi→Predmet →OLE predmet...". V pogovornem oknu lahko nato izberemo vrsto dokumenta (program), s katerim bomo ustvarili nov predmet. Lahko pa izberemo predmet iz datoteke, katere vsebina bo kot predloga za ustvarjanje novega predmeta.

3.8.3 Pogoste kombinacije tipk

Ustvari nov dokument	<Control>+<N>
Odpri dokument	<Control>+<O>
Natisni	<Control>+<P>
Shrani dokument	<Control>+<S>
Prikaz čez cel zaslon	<Control>+<Shift>+<J>
Pomoč	<F1>
Izberi vse	<Control>+<A>
Kopiraj	<Control>+<C>
Prilepi	<Control>+<V>
Izreži	<Control>+<X>
Razveljavi	<Control>+<Z>
Najdi in zamenjaj	<Control>+<F>

Tabela 3.1: Pogoste kombinacije tipk

Pogoste kombinacije tipk so navedene v tabeli 3.1.

3.9 Zaključek

Računalniki postajajo vse manjši, vgrajeni so v vse številnejše vsakdanje predmete, celo v obleke. Komuniciranje s takimi računalniki mora biti čim bolj intuitivno in naravno, kar pomeni predvsem razumevanje govora in slikovnih informacij. Drobni, v vsakdanje predmete vgrajeni in med seboj povezani računalniki naj bi omogočili tako imenovano okoljsko

inteligenco (angl. *ambient intelligence*), v kateri računalnike – razen po njihovih rezultatih – sploh ne bomo več opazili.

3.10 Naloge

1. Ali je možno oblikovati uporabniški vmesnik, ki bo ustrezal vsem vrstam uporabnikov?

2. Naštej štiri osnovne vrste komuniciranja z računalnikom in kakšna oprema je za to potrebna?

3. Kako pridobimo uporabniški model vmesnika?

4. Kaj bolj osrečuje uporabnika, veliko ali majhno število možnih izbir v določeni situaciji?

5. Zakaj uporabniki ne berejo navodil?

3.11 Koristne spletne povezave

1. Galerija grafičnih uporabniških vmesnikov:
   `http://toastytech.com/guis/indexlinks.html`

2. Napotki za izdelavo vmesnikov podjetja Apple:
   `http://developer.apple.com/documentation/UserExperience/Conceptual/OSXHIGuidelines/`

3. Napotki GNOME za izdelavo vmesnikov:
   `http://developer.gnome.org/projects/gup/hig/`

4. Napotki KDE za izdelavo vmesnikov:
   `http://developer.kde.org/documentation/standards/kde/style/basics/`

5. Spletna stran OpenOffice.org:
   `http://www.openoffice.org/`

6. Spletna stran slovenskega OpenOffice.org:
   `http://sl.openoffice.org/`
   (S te strani je pod licenco GNU/FDL dostopna tudi knjiga [4].)

7. OpenOffice.org zavezništvo – koristne informacije glede namestitve in uporabe paketa OpenOffice.org:
   `http://ooz.agenda.si/`

3.12 Literatura

[1] R. Ludvik, I. Zajc, and A. Medic. *Hitri vodnik po OpenOffice.org.* Pasadena, Ljubljana, 2003. (Dostopna pod licenco GNU/FDL na: http://openoffice.lugos.si/knjiga/Hitri_vodnik_po_OpenOffice.org_FDL.pdf).

[2] J. Raskin. *The Humane Interface, New Directions for Designing Interactive Systems.* Addison-Wesley, Reading, MA, 2000.

[3] E. S. Raymond. *The Art of Unix Programming.* Addison-Wesley Pub. Co., 2003.

[4] B. Shneiderman. *Designing the User Interface.* Addison-Wesley, Reading, MA, second edition, 1992.

[5] J. Spolsky. *User Interface Design For Programmers.* APress, 2001.

[6] E. R. Tufte. *The Visual Display of Quantitative Information.* Graphics Press, Cheshire, CT, 1983.

[7] E. R. Tufte. *Envisioning Information.* Graphics Press, Cheshire, CT, 1990.

[8] E. R. Tufte. *Visual Explanations.* Graphics Press, Cheshire, CT, 1997.

4

Operacijski sistemi

Operacijski sistem je poseben program, ki upravlja z delovanjem računalnika in skrbi za uporabniški vmesnik, organizacijo datotek na pomnilnih enotah in za nadzor nad vhodno-izhodnimi enotami. Zaradi svoje narave je tesno povezan s strojno opremo računalnika.

Kriteriji, po katerih delimo operacijske sisteme, so naslednji:

število uporabnikov: enouporabniški ali večuporabniški sistemi,

število hkratnih procesov: enoopravilni ali večopravilni sistemi,

vrste obdelav: paketne, interaktivne, paralelne, v realnem času.

uporabniški vmesnik:

- ukazni (MS-DOS, Unix),
- grafični (Macintosh OS, X Window System, MS Windows, OS/2),

strojna platforma:

- na različnih platformah (Unix),
- samo na istovrstni strojni platformi (npr. Microsoft Windows na PC kompatibilnih računalnikih),
- možna pa je programska emulacija operacijskega sistema (npr. MS Windows na računalnikih Macintosh).

cena:

- brezplačni (Linux),
- plačljivi (MS Windows).

Z operacijskim sistemom je tesno povezana tudi druga sistemska programska oprema, kot so prevajalniki za posamezne programske jezike in servisni programi, kot so na primer programi za urejanje datotek (angl. *editor*), programi za razvrščanje podatkov v datotekah, komunikacijski programi itd.

4.1 Linux

Linux je različica operacijskega sistema Unix, ki je postala v zadnjih letih še posebej popularna. Prve različice Unixa so bile razvite že pred desetletji, ko so ga kot raziskovalni operacijski sistem preizkušali in razvijali na univerzah. V začetku 80ih se je močno razširil z visoko zmogljivimi Sunovimi delovnimi postajami. Ko so v konkurenčni boj s Sunom na trgu delovnih postaj vstopila še ostala podjetja (HP, IBM, Silicon Graphics, Apollo ...), je vsako podjetje razvilo svojo različico operacijskega sistema Unix. Ker te niso bile popolnoma združljive, je to v veliki meri otežilo prodajo programske opreme. Z vstopom Microsofta na trg delovnih postaj je že kazalo, da bo odsotnost centralne avtoritete v svetu Unixa oslabila njegov položaj.

Takrat pa je na to področje vstopil Linux [3] in pritegnil ogromno pozornosti. Linus Torvalds, ki je razvil jedro operacijskega sistema, je izvorno kodo v celoti objavil in omogočil njeno brezplačno razširjanje. Povabil je tudi ostale razvijalce, da prispevajo svoj delež k jedru, pri čemer je postavil en sam pogoj: izvorna koda mora ostati javna in brezplačna. Tisoči programerjev so pričeli izboljševati Linux, kar je povzročilo njegov bliskovit razvoj. Obstaja več različnih distribucij Linuxa (Fedora, Suse, Mandrake, Debian, ...), ki so preko interneta dostopne brezplačno, kupiti pa se jih da tudi na zgoščenkah. Cena teh zgoščenk pokriva predvsem stroške medija in distribucije.

Na Unixu temelji tudi novi Macintoshev operacijski sistem OS X, ki je nadgrajen s preverjenim Macintoshevim grafičnim uporabniškim vmesnikom.

V nadaljevanju poglavja si bomo bolj podrobno ogledali Linux kot primer operacijskega sistema z ukaznim uporabniškim vmesnikom in tudi grafični način dela z Linuxom. Začnimo pa z živim Linuxom, ki se na priloženi zgoščenki imenuje Slix (navodila za delo z zgoščenko so zapisana v dodatku A).

4.2 Živi Linux

Zadnje čase so precej popularne tako zgoščenke z živim Linuxom. Na teh CD ploščah so shranjeni operacijski sistem Linux in aplikacije. Sistem

poženemo kar s plošče, saj nameščanje na trdi disk ni potrebno. To je izredno dobra lastnost, saj netehnični uporabniki načeloma silno neradi nameščajo in konfigurirajo sistem. Živi Linux jim omogoča, da se takoj lotijo dela, ne da bi se ukvarjali z nameščanjem in konfiguriranjem operacijskega sistema.

Živi Linux ima nekaj pomembnih primerov uporabe:

1. Nedestruktivno testiranje: če je uporabniku sistem všeč in če sploh deluje na njegovi strojni opremi.
2. Omrežni terminal: veliko nalog, ki jih uporabniki želijo opraviti je možno opraviti preko omrežnih storitev, še posebej prek spleta. Živi Linux lahko uporabljamo kot odličen omrežni terminal.
3. Vključene aplikacije: na živi Linux lahko gledamo kot na celovito aplikacijo. Tako lahko najdemo izvedbe, ki so posebej namenje določenim uporabam, npr. obdelavi besedila
4. Distribucija digitalnih vsebin: CD skupaj z digitalnimi vsebinami (dokumenti, video, zvok, demostracijski programi) vsebuje programe za uporabo in njihovo predvajanje. Tako lahko digitalne vsebine uporabljamo bolj neodvisno, saj spreminjanje zunanje programske opreme na uporabo vsebin manj vpliva in zato ne zastarajo tako hitro.
5. Namestitev: celovit delujoč sistem je za namestitev na trdi disk načeloma lažje obvladljiv kot razni primitivni programi za namestitev.
6. Servis – živi Linux lahko vsebuje programe za diagnostiko in servis strojnih komponent računalnika.

Poganjanje sistema neposredno z zgoščenke ima nekaj zanimivih lastnosti:

1. Sistema se ne da pokvariti, ker ga ni možno spreminjati. Vsak zagon je videti kot sveža namestitev sistema. Če ta deluje prvič, deluje tudi naslednjič. Seveda pa se uporabniška seja vseeno lahko nepravilno zaključi.
2. Uporaba živega Linuxa načeloma ne spreminja in ne zbriše podatkov na uporabnikovem trdem disku. Uporabniki se namreč silno bojijo izgube svojih podatkov.

Verjetno najbolj znan živi Linux sistem je Knoppix, ki ga je razvil Klaus Knopper na osnovi Linux-ove distribucije Debian. Sicer pa obstaja tudi

precej drugih izvedb, ki se tipično razlikujejo po podprtih jezikih in vključenih aplikacijah.

Obstaja tudi nekaj živih Linuxov, ki so posebej prilagojeni slovenskim uporabnikom. Posebej velja omeniti Slo-Tech Linux, ki so ga razvili v skupini Slo-Tech, in Slix, ki so ga razvili v Ljudmili.

Živi CD-ji so odlična priložnost za raziskovanje uporabniške izkušnje, saj naj bi idealno podpirali čim bolj neproblematično in enostavno uporabo končnemu uporabniku. Na podoben način pravzaprav deluje programska oprema za igralne konzole, npr. Sony PlayStation, ki je namenjena najširši publiki.

4.3 Linux – terminalski način dela

4.3.1 Prijava v sistem

Unix je večuporabniški, večopravilni, mrežni operacijski sistem. Omogoča delo z mrežo in delo na drugih računalnikih v lokalnem in globalnem omrežju. Za zagotavljanje varnosti in zasebnosti se mora uporabnik pred pričetkom dela v sistem prijaviti. Vsakemu uporabniku je dodeljeno uporabniško ime (angl. *username*), ki uporabnika enolično določa, za zaščito svojih podatkov pa dobi uporabnik tudi geslo (angl. *password*), ki ga lahko kasneje sam spreminja.

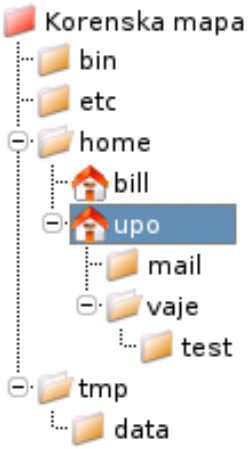
Prijava poteka tako, da vpišemo svoje uporabniško ime in nato še geslo. Pokaže se uporabniški vmesnik, ki je odvisen od tipa računalnika in nastavitev sistema. V splošnem ločimo med grafičnim in tekstovnim (terminalskim, ukaznim) uporabniškim vmesnikom. Ukazi za delo v terminalskem načinu se v različnih izvedbah operacijskega sistema Unix bistveno ne razlikujejo. V sistemu Slix je uporabnikova prijava samodejna s privzetim uporabniškim imenom.

4.3.2 Datotečni sistem

Datotečni sistem je v Unixu organiziran hierarhično (drevesno). Drevesna struktura datotečnega sistema se začne na korenskem imeniku (angl. *root*), ki je označen z znakom `/`. Enak znak se uporablja tudi kot ločilo pri opisovanju sestavljenih poti, na primer `test/vaja1.txt`. Z eno piko (`.`) je označen trenutni imenik, z dvema (`..`) pa imenik, ki se nahaja en nivo bližje korenskemu imeniku.

Diski in ostale vrste enot zunanjega pomnilnika so v Unixu vidni kot posamezni imeniki ali datoteke znotraj drevesne strukture datotečnega sistema. Ukaz `df` poleg ostalih infomacij izpiše tudi, kam (na kateri imenik) je posamezen disk nameščen (angl. *mounted*).

Ločimo relativno in absolutno opisovanje poti. Poti, ki se začnejo z znakom `/`, so absolutne. Relativne poti se ne začnejo z znakom `/`, označujejo pa relativen položaj datotek in imenikov glede na trenutni imenik.



Slika 4.1: Primer drevesne strukture imenikov

Primer: Nahajamo se v imeniku `upo`, katerega položaj v datotečnem sistemu je prikazan na sliki 4.1. S potjo `/tmp/data` je označen imenik `data`, ki se nahaja v imeniku `tmp`, slednji pa se nahaja v korenskem imeniku `/`. To je primer absolutne poti. Z relativno potjo `vaje/test` je označen imenik `test`, ki se nahaja v imeniku `vaje`, ta pa se nahaja v imeniku, kjer se trenutno nahajamo (`/home/upo`). Relativni naslov poti je tudi `../bill`, ki označuje imenik `bill`, ki se nahaja v imeniku `home`.

Lastništvo datotek, tipi datotek in atributi

Poglejmo primer izpisa ukaza `ls -l`, s katerim izpišemo vsebino trenutnega imenika (ukaz `ls` bo opisan kasneje).

```
drwx------   2 upo     mail   1024 Sep 30 13:48  Mail/
drwxrwxrwx   2 upo     vaje   1024 Jul 28 10:04  bin/
-rw-rw-rw-   1 upo     vaje  75144 Jul 20 13:47  data.txt
drwxrwxrwx   3 upo     vaje   1024 Oct 27 13:56  ftp/
-rw-rw-rw-   1 upo     vaje 154620 Oct 21 11:19  mbox
drwxrwxrwx   3 upo     vaje   1024 Jun 30 21:15  public_html/
drwxrwxrwx   5 upo     vaje   1024 Mar 24  1997  www/
```

Kot lahko vidimo v 3. in 4. stolpcu izpisa, ima vsaka datoteka svojega lastnika in uporabniško skupino, ki ji pripada. Npr. datoteka `data.txt` pripada uporabniku `upo` in skupini `vaje`.

V prvem stolpcu izpisa vidimo, da ima vsaka datoteka tudi 10 atributov, ki določajo njeno vrsto ter dovoljenja za dostop. Vidimo jih kot niz desetih znakov, v katerem ima vsak znak svoj pomen. Prvi določa vrsto datoteke. Vrste datotek so prikazani v tabeli 4.1. Naslednjih 9 znakov predstavlja 3 skupine s po tremi znaki. Skupina prvih treh določa dovoljenja za dostop do datotek, ki jih ima lastnik datoteke. Druga skupina določa dovoljenja za pripadnika iste skupine (kot lastnik datoteke), zadnji trije znaki pa dovoljenja za vse ostale, kot je to prikazano na sliki 4.2. Tabela 4.2 prikazuje pomen znakov znotraj vsake skupine.

4.3.3 Delo z ukazi lupine operacijskega sistema

Lupina operacijskega sistema (angl. *OS shell*) je ukazni tolmač (angl. *interpreter*), ki služi kot vmesnik med uporabnikom in jedrom operacijskega sistema. V nadaljevanju bodo opisani osnovni ukazi za delo z operacijskim sistemom Linux iz terminalskega okna. Ukaz prikličemo z imenom ukaza, ki mu po potrebi dodamo stikala in dodatne argumente, ločene s presledkom. Za stikala se uporablja pomišljaj (`-`). Neobvezni deli ukaza so v oglatih oklepajih (`[` in `]`), izključujoča se stikala pa so ločena z znakom ali (`|`).

---

man

Z ukazom `man` izpišemo pomoč za določeno ključno besedo ali ukaz iz priročnika. Če se opis nahaja v več poglavjih v priročniku, lahko navedemo tudi številko poglavja.

Format:
`man [poglavje] ključna_beseda`

Navodila za uporabo posameznih ukazov so navadno zelo obširna, prikaz pa je razdeljen po straneh. V zadnji vrstici lupina pričakuje uporabnikov ukaz. Z `<Enter>` pomaknemo prikaz za eno vrstico naprej, s preslednico pa za celoten ekran. Prikaz se prekine, ko izpišemo še zadnjo vrstico pomoči, med pregledovanjem pa ga lahko ustavimo s `q`.

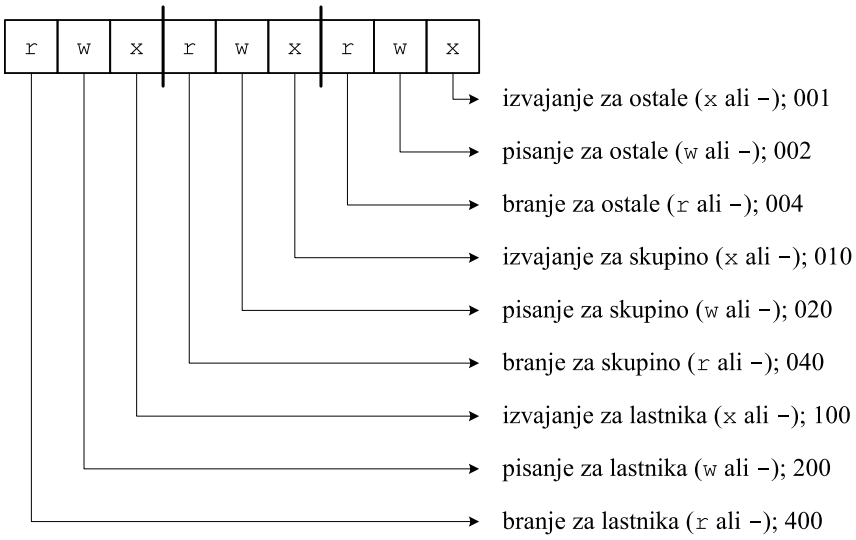
Krajšo pomoč o uporabi ukaza prikažemo s stikalom `--help`, s katerim izpišemo kratek opis stikal, ki jih lahko uporabimo z ukazom.

Primer: izpis pomoči za ukaz `ls`:

```
man ls
ls --help
```

Znak	*Tip datoteke*
`-`	navadna datoteka (angl. *file*)
`d`	imenik (angl. *directory*)
`l`	povezava (angl. *link*)
`b`	posebna blokovna datoteka (angl. *block special file*)
`c`	posebna znakovna datoteka (angl. *character special file*)
`p`	posebna cevna datoteka (angl. *named pipe special file - FIFO*)
`n`	mrežna posebna datoteka (angl. *network special file*)
`s`	vtičnica (angl. *socket*)
`H`	skriti imenik

Tabela 4.1: Vrste datotek v Linuxu



Slika 4.2: Shematski prikaz dovoljenj za dostop do datoteke

Znak	*Pomen*
`-`	ni dovoljenja
`r`	dovoljenje za branje
`w`	dovoljenje za pisanje
`x`	dovoljenje za izvajanje ali iskanje po imeniku

Tabela 4.2: Dovoljenja za dostop in njihove oznake

&

Večopravilni način delovanja (angl. *multitasking*).

Če ukazu sledi znak & (angl. *and*), se ukaz izvede kot nov proces in lupina OS se takoj vrne v ukazno vrstico. V nasprotnem primeru je potrebno počakati, da se ukaz izvrši do konca. Ukaz je še posebej uporaben v okenskem načinu, ko iz enega terminalskega okna zaženemo več aplikacij (npr. mozilla, xdvi, gv, emacs) in pri tem nemoteno nadaljujemo delo.

Primer:

```
mozilla &
emacs teta.tex &
xdvi teta.dvi &
gv teta.ps &
```

Kombinacija tipk <CTRL>+<C>

Včasih se primeri, da izvajanje določenega ukaza zmrzne. Če ukaza nismo izvedli v večopravilnem načinu, nam takšna težava onemogoči nadaljnje delo, saj se lupina OS ne vrne v ukazno vrstico. V tem primeru lahko izvajanje ukaza prekinemo s kombinacijo tipk <CTRL>+<C>. Če zmrzne ukaz, ki ga poženemo v večopravilnem načinu dela kot nov proces, pa ga je potrebno končati z ukazom `kill`, ki bo opisan v nadaljevanju.

4.3.4 Osnovni ukazi za delo z datotečnim sistemom

ls

Izpis vsebine imenika in podatkov o datotekah.

Format:
`ls [-adltxCR] [imena ...]`

Stikala:

- `a` Izpis vseh datotek (angl. *all*).
- `d` Če je argument imenik, se izpiše samo njegovo ime in ne vsebina.
- `l` Izpis v dolgem formatu (ime, dovoljenja, lastnik, skupina, dolžina v bajtih, čas zadnje spremembe).
- `t` Izpis, urejen po času zadnje spremembe.
- `x` Večstolpčni izpis z imeni, urejenimi po vrsticah.
- `C` Večstolpčni izpis z imeni, urejenimi po stolpcih.
- `R` Rekurziven izpis vsebine naštetih imenikov.

Primer: daljši izpis vsebine domačega imenika uporabnika “miha”:

```
ls -al ~miha
```

---

cd

Spreminjanje delovnega (trenutnega) imenika.

Format:
`cd [imenik]`

Pri sklicevanju na imenike lahko uporabljamo tri okrajšave: `.` predstavlja trenutni imenik, `..` starševski imenik, ki se nahaja v drevesni strukturi en nivo višje, `~uporabniško_ime` pa uporabnikov domači imenik (npr. `~upo`); če ne navedemo uporabniškega imena, se prestavimo v svoj domači imenik. V svoj domači imenik se prestavimo tudi, če ne navedemo imenika.

Primer: pomik na domači imenik uporabnika “miha”:

```
cd ~miha
```

---

cp

Kopiranje datotek in imenikov.

Format:
```
cp [-f | -i] [-p] datoteka nova_datoteka
cp [-f | -i] [-p] datoteka1 [datoteka2 ...]  ciljni_imenik
cp [-f | -i] [-p] [-r] imenik1 [imenik2 ...]  ciljni_imenik
```

Stikala:

- `f` Brez vprašanja prepiše že obstoječe datoteke z novimi.
- `i` Če ciljna datoteka že obstaja, vpraša, ali naj jo prepiše ali ne.
- `p` Ohrani dovoljenja, lastnika in skupino, kateri pripada datoteka.
- `r` Rekurzivno skopira celotno drevesno strukturo.

Primer: kopiranje tekstovnih datotek iz imenika `~miha/download` v trenutni imenik:

```
cp ~miha/download/*.txt .
```

---

mv

Premikanje, preimenovanje datotek in imenikov.

Format:
```
mv [-f | -i] datoteka nova_datoteka
mv [-f | -i] datoteka1 [datoteka2 ...]  ciljni_imenik
mv [-f | -i] imenik1 [imenik2 ...]  ciljni_imenik
```
Stikala:

- `f` Brez vprašanja prepiše že obstoječe datoteke z novimi.
- `i` Če ciljna datoteka že obstaja, vpraša, ali naj jo prepiše ali ne.

Primer: premik izvornih datotek dokumenta v LaTeX-u iz trenutnega v imenik, ki je nivo bližje korenskemu imeniku. Če v ciljnem imeniku že obstaja kakšna datoteka z istim imenom, bo brez vprašanja prepisana:

```
mv -f *.tex ..
```

---

rm

Brisanje datotek in imenikov.

Format:
```
rm [-f | -i] [-r] datoteka ...
```
Stikala:

- `f` Datoteke zbriše brez vprašanja.
- `i` Za vsako datoteko moramo potrditi njeno brisanje.
- `r` Preden zbriše imenik, rekurzivno zbriše vso njegovo vsebino.

Primer: brisanje vsebine trenutnega imenika z vsemi podimeniki:

```
rm -r *
```

---

pwd

Izpis imenika, v katerem se nahajamo.

Format:
```
pwd
```

---

mkdir

Ustvarjanje imenika.

Format:

`mkdir [-p] [-m dovoljenja] nov_imenik`

Stikala:

- `p` Po potrebi se ustvari vmesna pot do ciljnega imenika. Npr., če želimo narediti imenik `vaje/test/podatki` in imenik `vaje` še ne obstaja, se najprej naredi imenik `vaje`, nato imenik `test` in nato še imenik `podatki`.
- `m` Na novo narejeni imenik bo imel dovoljenja nastavljena na vrednosti, ki so v parametru dovoljenja (glej poglavje 4.3.2).

Primer: v trenutnem imeniku bomo ustvarili gnezdene imenike `vaje/test/podatki`:

```
mkdir -p vaje/test/podatki
```

---

rmdir

Brisanje praznega imenika.

Format:
`rmdir [-p] imenik`

Stikala:

- `p` Če je po brisanju imenika s sestavljenim imenom (npr. `vaje/test` ima 2 dela: `vaje` in `test`) imenik na višjem nivoju prazen, se pobriše tudi ta.

Primer: brisanje praznega imenika `temp`:

```
rmdir temp
```

---

df

Izpis informacij o nezasedenem prostoru na diskih (datotečnih sistemih).

Format:
`df [-h]`

Stikala:

- `h` Velikosti se izpišejo v prijaznejši obliki (npr. 1K, 230M, 2G).

Primer: izpis informacije o zasedenosti particije, na kateri se nahajamo:

```
df .
```

---

du

Izpis prostora, ki ga na disku zavzema imenik ali datoteka.

Format:
`du [-a | -s] [-m | -k] [datoteka | imenik ...]`

Stikala:

- `a` Izpiše se tudi informacija o vsaki datoteki v imeniku.
- `s` Izpiše se samo skupen prostor, ki ga na disku zaseda vsaka navedena datoteka ali imenik.
- `k` Izpis dolžine v KB.
- `m` Izpis dolžine v MB.

Primer: izpis prostora (v MB), ki ga zasedajo vse datoteke v trenutnem imeniku:

```
du -sm .
```

4.3.5 Ukazi za delo z datotekami

find

Iskanje datotek in imenikov.

Format:
`find imenik [-name ime] [-type tip_datoteke] [-print]`

Stikala:

- `name` Ime datoteke, ki jo želimo najti.
- `type` Tip datoteke, ki jo želimo najti. `tip_datoteke` je eden izmed tipov datotek, ki so navedeni v tabeli 4.1, z eno razliko: navadna datoteka je tukaj označena z znakom `f` in ne z znakom `-`.
- `print` Izpiše ime, ki ustreza iskalnim pogojem.

Primer: Poiščimo datoteke MP3 na domačem imeniku trenutno prijavljenega uporabnika:

```
find ~ -name *.mp3 -print
```

---

ln

Ustvarjanje povezav (angl. *link*) do datotek in imenikov.

Format:

```
ln [-f] [-i] [-s] datoteka nova_datoteka
ln [-f] [-i] [-s] datoteka1 [datoteka2] ciljni_imenik
ln [-f] [-i] [-s] imenik1 [imenik2] ciljni_imenik
```

Stikala:

- `f` Če ciljna datoteka že obstaja, jo brez vprašanja prepiše.
- `i` Za vsako ciljno datoteko, ki že obstaja, smo vprašani, ali jo želimo prepisati.
- `s` Namesto običajne trde povezave se ustvari simbolna (mehka) povezava.

Funkcionalno so trde in simbolne povezave podobne, a določene razlike obstajajo. Do imenika ali do datoteke v drugem datotečnem sistemu lahko naredimo samo simbolno povezavo. Simbolne povezave so nam v pomoč, ker določajo datoteko, na katero kažejo; ta se pri ukazu `ls` tudi izpiše. Pri trdih povezavah pa ni preprostega načina, da bi ugotovili, katere datoteke so povezane.

Primer: ustvarimo najprej mehko povezavo do datoteke `/etc/fstab`, nato pa poglejmo, kam kaže ustvarjena povezava:

```
ln /etc/fstab testLink
ls -l testLink
```

---

more

Izpis vsebine datoteke.

Format:

```
more datoteka
```

Primer: izpišimo vsebino datoteke, na katero kaže povezava `testLink`:

```
more testLink
```

---

tail

Izpis repa datoteke.

Format:

```
tail [-c N] datoteka
tail [-n N] datoteka
tail [-f] datoteka
```

Stikala:

- `c` Izpiše zadnjih N znakov datoteke.

`n` Izpiše zadnjih N vrstic datoteke.
`f` Aktivno spremlja zadnjih nekaj vrstic datoteke, tudi ko se ta spreminja.

Primer: izpišimo zadnji dve vrstici datoteke, na katero kaže povezava `testLink`:

```
tail -n2 testLink
```

---

wc

Štetje znakov, besed in vrstic po datotekah.

Format:
`wc [-lwc] [datoteke]`

Stikala:

`l` Izpis števila vrstic.
`w` Izpis števila besed.
`c` Izpis števila znakov.

Če ne navedemo datotek, se štetje izvrši na standardnem vhodu. Primer standardnega vhoda je tipkovnica. Če ne navedemo stikal, se izpišejo vsi trije podatki.

Primer: preštejmo besede v povezavi `testLink`:

```
wc -w testLink
```

4.3.6 Tiskanje

lpr

Izpis datoteke na tiskalnik.

Format:
`lpr [-Ptiskalnik] [-m] [-#št_kopij] datoteka`

Stikala:

`P` Izberemo tiskalnik, na katerem želimo izpis. V nasprotnem primeru se uporabi privzeti tiskalnik.
`m` Ko so datoteke natisnjene, dobimo obvestilo po elektronski pošti.
`#` Izpiše datoteko v `št_kopij` kopijah.

Primer: tiskanje datoteke `/etc/fstab`:
`lpr textLink`

---

lpq

Prikaz opravil, ki čakajo na tiskanje, in prikaz stanja določenega tiskalnika ali celotnega sistema za tiskanje. Zahtevamo lahko tudi prikaz stanja točno določenih opravil (`št_opravila`) ali vseh opravil samo za določene uporabnike (`uporabniki`).

Format:
`lpq [-l] [-Ptiskalnik] [št_opravila] [uporabniki]`

Stikala:

- `l` Izpis informacij o vsaki datoteki, ki sestavljajo opravilo v vrsti.
- `P` Izpis statusa navedenega tiskalnika. Če to stikalo izpustimo, se uporabi privzeti tiskalnik.

`lpq` brez vseh argumentov prikaže vsa opravila, ki so trenutno v vrsti za tiskanje.

---

lprm

Brisanje opravil (`št_opravila`), ki čakajo na tiskanje, ali vseh opravil določenih uporabnikov (`uporabniki`). Številke opravil dobimo z ukazom `lpq`.
Format:
`lprm [-Ptiskalnik] [-] [št_opravila] [uporabniki]`

Stikala:

- `P` Brisanje opravil navedenega tiskalnika. Če to stikalo izpustimo, se uporabi privzeti tiskalnik.

Z `lprm -` izbrišemo vsa svoja opravila, ki še čakajo na tiskanje.

4.3.7 Ukaza za spreminjanje dovoljenj in lastništva datotek

chmod

Spreminjanje dovoljenj datotek in imenikov.

Format:
`chmod [-R] dovoljenja datoteke`

Stikala:

- `R` Rekurzivno spremeni dovoljenja. Za vsak navedeni imenik se dovoljenje rekurzivno nastavi vsem datotekam in imenikom v njem.

Dovoljenja so lahko številčna ali simbolična:

- Številčna dovoljenja so predstavljena kot števila v osmiškem številskem sistemu, v katerih posamezen bit pomeni določeno dovoljenje (glej sliko 4.2.). Številčna dovoljenja so vedno absolutna. Npr. 754 pomeni dovoljenje za branje, pisanje in izvajanje za lastnika, dovoljenje za branje in izvajanje za skupino in dovoljenje za branje za ostale. Če 754 zapišemi binarno, dobimo 111 101 100, kar ustreza `rwxr-xr--`.

- Simbolična dovoljenja so lahko absolutna ali relativna. Absolutna dovoljenja so neodvisna od obstoječih dovoljenj datoteke, relativna pa obstoječa dovoljenja dodajajo ali odvzemajo. Simbolična dovoljenja imajo obliko seznama elementov, ki so ločeni z vejicami. Vsak element ima obliko:

  `[<kdo>]<operator>[<dovoljenje>]`

  `<kdo>` pomeni, na koga se to dovoljenje nanaša, in je lahko poljubna kombinacija naslednjih vrednosti:
  - `u` Spremeni dovoljenja za lastnika.
  - `g` Spremeni dovoljenja za skupino.
  - `o` Spremeni dovoljenja za ostale.
  - `a` Spremeni dovoljenja za vse (je enako kot `ugo`).

  `<operator>` mora biti naveden. Možni operatorji so:
  - `+` Dovoljenje se doda.
  - `-` Dovoljenje se odvzame.
  - `=` Dovoljenje se nastavi.

  `<dovoljenje>` pa je poljubna kombinacija sledečih vrednosti:
  - `r` Dovoljenje za branje.
  - `w` Dovoljenje za pisanje.
  - `x` Dovoljenje za izvajanje/iskanje.
  - `X` Če je datoteka imenik, potem to pomeni dovoljenje za iskanje. Če je datoteka imenik ali dovoljenja datoteke vsebujejo dovoljenje za izvajanje/iskanje za vsaj enega od dovoljenj lastnika, skupino ali ostale, se obnaša tako kot x. Če datoteka ni imenik in ne vsebuje dovoljenj za izvajanje, ne naredi nič.

Primer: dovolimo vsem uporabnikom skupine branje in spreminjanje datoteke test.txt:

```
chmod g+rw test.txt
```

Primer: dovolimo lastniku branje in pisanje, skupini le pisanje, ostalim uporabnikom pa ne dovolimo dostopa do tekstovnih datotek na trenutnem imeniku:

```
chmod 620 *.txt
```

---

chown

Spreminjanje lastništva datotek in imenikov. *Format:*
`chown [-R] lastnik[:skupina] datoteke`

Stikala:

`R` Če je datoteka imenik, se lastništvo rekurzivno spremeni za vse datoteke in podimenike v njem.

Primer: dodelimo datoteko `test.txt` uporabniku "miha":

```
chown miha test.txt
```

4.3.8 Ostali osnovni ukazi

date

Izpis in nastavljanje sistemskega datuma in ure.

Format:
`date [MMddhhmm[[cc]yy][.ss]]`

Ukaz brez parametrov prikaže trenutni datum in čas. Za nastavljanje ure in datuma potrebujemo administratorske privilegije.

Znaki v nizu `MMddhhmm[[cc]yy][.ss]` za nastavljanje časa pomenijo sledeče:

`MM` Mesec (01 do 12).
`dd` Dan (01 do 31).
`hh` Ura (00 do 23).
`mm` Minuta (00 do 59).
`cc` Stoletje.
`yy` Leto (00 do 99). Če leta ne navedemo, se upošteva tekoče leto.
`ss` Sekunda (00 do 59).

Primer: nastavitev sistemskega datuma in ure na 1. oktober, 13:00:

```
date 10011300
```

---

passwd

Spreminjanje uporabniškega gesla.

Format:
`passwd [uporabnik]`

Če uporabnika ne navedemo, spremenimo geslo trenutnega uporabnika.

---

who

Izpis vseh uporabnikov, ki so trenutno prijavljeni na sistem.

Format:
`who [-Hq]`
`who am I`

Stikala:

- `H` Izpiše se tudi vrstica z imeni posameznih polj.
- `q` Izpiše samo število uporabnikov in njihova imena.

`who am I` izpiše podatke o tem, kdo smo (uporabniško ime, ime linije in čas prijave v sistem).

---

ps

Izpis podatkov o aktivnih procesih.

Format:
`ps [-efl] [-t terminali] [-p št_procesov] [-u uporabniki] [-g skupine]`

Stikala:

- `e` Izpis podatkov o vseh procesih.
- `f` Za posamezen proces izpiše dodatne podatke.
- `l` Za posamezen proces izpiše vse podatke, ki so na voljo.
- `t` Izpis podatkov o tistih procesih, ki so povezani z navedenim terminalom.
- `p` Izpis podatkov samo za navedene procese.
- `u` Izpis podatkov o tistih procesih, ki jih poganjajo navedeni uporabniki.
- `g` Izpis podatkov o tistih procesih, ki jih poganjajo uporabniki iz navedenih skupin.

Primer: obširnejši izpis podatkov o vseh procesih:

`ps -el`

---

kill

Končamo (ubijemo) proces.

Format:
`kill PID`

`PID` je identifikacijska številka procesa, ki ga želimo končati. Dobimo jo z ukazom `ps`.

Primer: Denimo, da se je zaciklal brskalnik Mozilla. Najprej s `ps` izvemo identifikacijsko številko PID procesa. Proces končamo z ukazom `kill`:

```
ps
kill 989
```

---

telnet

Prijava na oddaljeni računalnik.

Format:
`telnet oddaljeni_računalnik [številka_vrat]`

`oddaljeni_računalnik` je ime ali številka IP računalnika, na katerega se želimo prijaviti. `številka_vrat` pomeni številko vrat (angl. *port number*), preko katerih se priključimo na oddaljeni računalnik. Privzeta številka vrat, ki je rezervirana za `telnet`, je 23.

Primer: prijava na računalnik `lrv.fri.uni-lj.si`:

```
telnet lrv.fri.uni-lj.si
```

---

rlogin

Prijava na oddaljeni računalnik.

Format:
`rlogin oddaljeni_računalnik [-l uporabniško_ime]`

Stikala:

- `l` Določimo uporabniško ime, s katerim se prijavimo na oddaljeni računalnik. Če ga ne navedemo, se privzame uporabniško ime, s katerim smo trenutno prijavljeni v sistem.

Primer: prijava na računalnik `lrv.fri.uni-lj.si` z uporabniškim imenom `miha`:

```
rlogin lrv.fri.uni-lj.si -l miha
```

---

ssh

Varna prijava na oddaljeni računalnik.

Format:
`ssh [-l uporabniško_ime] [-p številka_vrat] [-X] oddaljeni_računalnik`

Stikala:

- `l` Določimo uporabniško ime, s katerim se prijavimo na oddaljeni računalnik. Če ga ne navedemo, se privzame uporabniško ime, s katerim smo trenutno prijavljeni v sistem.
- `p` Določimo številko vrat, na katerih se na oddaljenem računalniku nahaja strežnik `SSH`. Privzeta številka vrat je 22.
- `X` Omogočimo posredovanje protokola X11, s čimer lahko uporabljamo grafične programe na oddaljenem računalniku tako, kot da bi jih izvajali lokalno (če je povezava dovolj hitra).

Primer: prijava na računalnik `giant.fri.uni-lj.si` z uporabniškim imenom `miha` in omogočenim posredovanjem grafičnega protokola X11:

```
rlogin -l miha -X lrv.fri.uni-lj.si
```

4.3.9 Delo z zunanjimi pomnilniškimi napravami

Kot je bilo razloženo v uvodnem podpoglavju, so vse datoteke, ki so dostopne v OS Unix, urejene v drevesno strukturo, katerega koren je korenski imenik (angl. *root*). Datoteke se lahko nahajajo na poljubni napravi (na primer na particiji trdega diska, disketi, CD-ROMu). Datotečni sistem torej logično združuje dostop do datotek, ki se fizično lahko nahajajo na različnih pomnilnih medijih. Delo z vsemi zunanjimi pomnilniškimi napravami (kot sta na primer disketna enota in enota CD-ROM) je podobno. Datotečni sistem naprave moramo najprej priklopiti (angl. *mount*) v datotečni sistem operacijskega sistema. To storimo z ukazom `mount`, ki prebere podatke (oz. del podatkov) v delovni (med)pomnilnik. Ko spreminjamo podatke (brišemo, spreminjamo in dodajamo nove datoteke) na priklopljeni enoti, se podatki spreminjajo v (med)pomnilniku. Zato moramo, preden končamo delo z zunanjo napravo, spremembe tudi dejansko zapisati nanjo. To storimo z ukazom `umount`, ki prenese vse spremembe iz medpomnilnika nazaj na pomnilni medij zunanje naprave.

Opis datotečnih sistemov zunanjih naprav, ki so že priklopljene ali jih lahko priklopimo v enotni datotečni sistem, se nahaja v datoteki `/etc/fstab`. V prvem stolpcu je oznaka naprave, v drugem pa imenik,

kamor se naprava priklopi. `/etc/mtab` vsebuje trenutno priklopljene naprave. Primer dela vsebine datoteke `fstab`:

```
/dev/fd0          /mnt/floppy       ext2    owner,noauto    0 0
/dev/hdc          /mnt/zip          vfat    noauto,user     0 0
/dev/cdrom        /mnt/cdrom        iso9660 owner,noauto,ro 0 0
```

mount in umount

Priklop/odklop naprave, ki je navedena v `/etc/fstab`, v datotečni sistem.

Format:
`mount imenik`
`umount imenik`

Priklopljeni datotečni sistem se nahaja na poti `imenik` - z napravo delamo enako kot z datotekami na tem imeniku. Ko delo končamo, jo odklopimo, s čimer se vsebina tudi dejansko prenese iz medpomnilnika na pomnilni medij.

Primer uporabe disketne enote: `mount /mnt/floppy`
```
cp mama.txt /mnt/floppy/mama.txt
cp *.tex /mnt/floppy
cp /mnt/floppy/ata.txt .
umount /mnt/floppy
```

4.3.10 Ustvarjanje nove datoteke

Standardni urejevalnik besedila na OS Unix je `vi`. Starosta urejevalnikov, ki deluje v terminalskem načinu, se nahaja na OS Unix vseh proizvajalcev. Delo v njem je za povprečnega uporabnika precej okorno, zato bomo raje predstavili urejevalnik emacs, ki je prav tako dovolj razširjen. Emacs zaženemo z ukazom[1]:

`emacs [imeDatoteke]`

Primer: `emacs mama.tex &`

Če navedemo tudi ime datoteke, program to datoteko naloži, v primeru, da datoteka še ne obstaja, program ustvari novo datoteko s tem imenom.

Uporabniški vmesnik urejevalnika je razdeljen na več delov. Na vrhu je menijska vrstica, ki jo najlažje pregledujemo z miško. Največji del okna je namenjen pisanju besedila. Pri pomikanju skozi besedilo si lahko pomagamo z drsnikom, ki je na levi strani okna. Spodnji vrstici zavzemata statusna vrstica ter vrstica, v katero vnašamo posebne ukaze namenjene urejevalniku.

[1] V primeru, da na priloženi zgoščenki Slix tega ukaza ne pozna, poskusite z ukazom `xemacs` ali pa z ukazom `Emacs`.

Nekaj najpomembnejših ukazov iz menijske vrstice:

"Files→Open File"	Odpremo novo datoteko.
"Files→Save Buffer"	Shranimo datoteko.
"Files→Save Buffer As"	Shranimo datoteko z drugim imenom.
"Buffers"	Pomikanje med odprtimi okni.
"Files→Kill Current Buffer"	Zapremo trenutno okno.
"Files→Exit Emacs"	Zaključimo delo.

4.4 Linux – grafični način dela

Danes si operacijskega sistema brez grafičnega uporabiškega vmesnika skoraj ne moremo več predstavljati. Do konca 80-tih let prejšnjega stoletja je na osebnih računalnikih kraljeval operacijski sistem Microsoft DOS (Disk Operating System), ki je po načinu delovanja z uporabniškega stališča zelo podoben terminalskemu načinu dela pod Linuxom. Ena izmed pomankljivosti operacijskega sistema DOS je bila tudi ta, da ni imel grafičnega uporabniškega vmesnika. Ta pomanjkljivost je postala še bolj očitna, ko je prišel na trg prvi Applov računalnik Macintosh. Microsoft je grafični uporabniški vmesnik uvedel s programom Microsoft Windows, ki pa tedaj še ni bil samostojen operacijski sistem. Pri razvoju svojih oken se je Microsoft očitno zgledoval po Macintoshevem grafičnem uporabniškem vmesniku, kar je povzročilo tudi sodne spore. V tem poglavju pa bo beseda tekla o grafičnem uporabniškem vmesniku operacijskega sistema Linux.

4.4.1 XWin in namizje

Kaj je XWin?

XWin (X Window System) je sistem, ki uporablja protokol X, ta pa omogoča grafično prikazovanje na računalnikih z operacijskim sistemom Unix oziroma njegovih različicah. To ni operacijski sistem, je le razširitev različic Unixa, torej tudi Linuxa.

Pri uporabi sistema XWin ima lahko uporabnik na zaslonu hkrati več oken. Pogosto se z XWin uporablja tudi kazalna naprava, na primer miška, vendar ta ni nujno potrebna.

XWin, ki je bil prvotno razvit na Massachusetts Institute of Technology (MIT), sodi v sklop odprtega programja (glej 2. poglavje). Veliko komercialnih proizvajalcev je razširjalo lastne izboljšave originalnega sistema XWin. Različica XWin, ki teče na Linuxu, se imenuje XFree86, prenos X11R6 (X Window System, različica 11, izdaja 6) na računalnike s procesorji Intel (80x86). XFree86 podpira širok razpon grafičnih kartic, grafičnih pospeševalnikov in video opreme. XFree86 je popolna

distribucija sistema XWin in vsebuje sam sistem oziroma strežnik XWin, veliko aplikacij in pripomočkov, programskih knjižnic in dokumentacije.

Aplikacij za XWin, ki tečejo pod Linuxom, je preveč, da bi jih tukaj omenjali, med njimi pa so programi za urejanje preglednic, urejevalniki besedil, grafični programi, spletni brkljalniki itd. Teoretično bi se morala vsaka aplikacija, napisana za XWin, povsem brez napak prevesti za Linux.

Uporabnik, ki uporablja Linux, sploh ne zazna, da grafičen uporabniški vmesnik deluje na osnovi sistema XWin, saj vidi le tipične gradnike vmesnika, s katerimi poteka interakcija.

Če Linux zaženemo v terminalskem načinu, grafičen način vklopimo tako, da v ukazni vrstici zaženemo ukaz `startx`, vendar se danes lahko Linux osnovnih nastavitvah zažene že tudi v grafičnem načinu.

Upravljanje oken

Sistem XWin nam torej omogoča osnovno grafično prikazovanje, ob namestitvi pa si moramo izbrati tudi samo namizje, ki definira delovno okolje[2]. Z vsakim namizjem se namesti tudi upravljalec oken (Window Manager).

Medtem ko upravljalec oken skrbi za postavljanje oken in njihov videz, namizje skrbi za vsebino teh oken.

Upravljalci oken nam nudijo naslednje osnovne operacije:

- postavljanje oken,
- ikonizacijo oziroma minimiziranje oken,
- večanje, manjšanje in prestavljanje oken,
- videz okenskih okvirjev,
- pomično namizje – ko z miško pridemo do roba zaslona, se namizje pomakne, kot da bi bilo precej večje, kot je v resnici[3],
- delovna področja ali navidezna namizja – več namizij, med katerimi preklapljamo z namenom, da imamo večji nadzor nad odprtimi okni.

KDE in GNOME

KDE in GNOME sta dve najbolj standardni namizji. Z namizjem KDE dobimo poleg še upravljalca oken KWM, z namizjem GNOME pa upravljalca oken Enlightenment. Bistvena razlika med obema namizjema

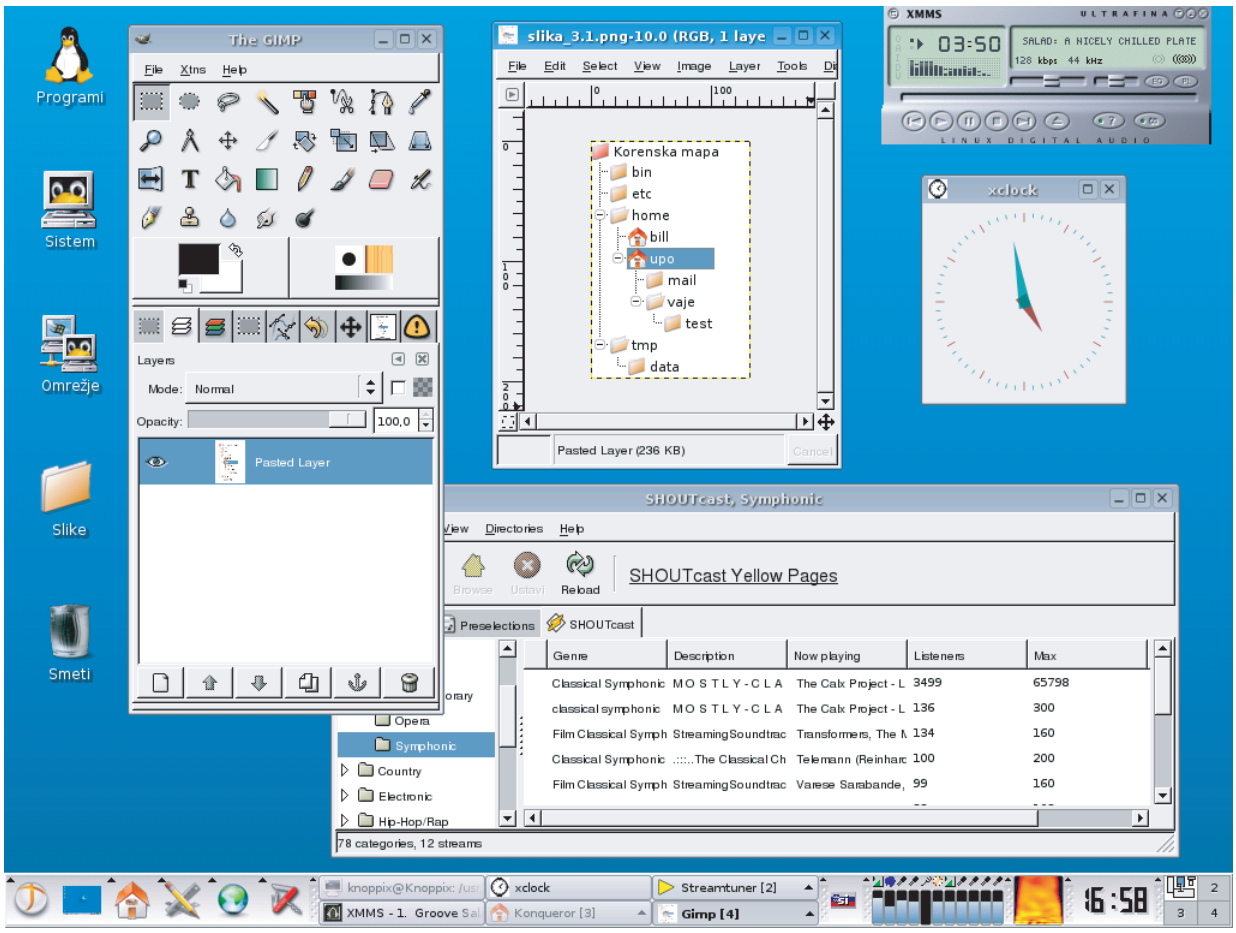
[2]Namizje oziroma delovno okolje zajema vse, kar vidimo na zaslonu.
[3]Velikost vidnega dela namizja določa izbrana ločljivost prikazovanja na zaslonu.

je, da je KDE predmetno usmerjen, GNOME pa ne. V samem videzu namizij pa ni bistvenih razlik.

Seveda pa obstaja še več namizij (na primer CDE) in še več upravljalcev oken (FVWM, FVWM95, MWM, Afterstep, WindowMaker, SawMill, BlackBox, itd.), ki si jih lahko uporabnik namesti tudi, če ni skrbnik (angl. *root*) računalnika.

V nadaljevanju bomo obravnavali namizje KDE 3.3.

4.4.2 Pomen ikon na pultu in namizju



Slika 4.3: Primer Linuxovega namizja KDE 3.3

Na sliki 4.3 vidimo *primer* Linuxovega namizja KDE 3.3. Oglejmo si bistvene lastnosti namizja, ki so razvidne s te slike:

- V skrajnem levem kotu pulta imamo izhodiščni gumb, ki služi kot

dostop do menijske strukture, kjer najdemo opcije za delo - od pomoči do aplikacij. Več o opcijah, ki nam jih ponuja ta gumb, bomo spregovorili v naslednjem poglavju.

- S pritiskom na tretji gumb odpremo meni, kjer lahko izbiramo med upravljalcem datotek, terminalskim oknom, dokumenti, s katerimi smo nazadnje delali, ter sistemskimi aplikacijami.

- S pritiskom na četrti gumb odpremo meni, kjer se nahajajo uporabniške aplikacije, kot so urejevalniki besedil, grafični programi, programi za prezentacije in multimedijske aplikacije.

- Peti gumb ponuja medmrežne aplikacije, kot so brskalniki, poštni odjemalci, programi za dostop do oddaljenih računalnikov, ipd.

- S pomočjo šestega gumba na pultu pa pridemo do raznih pripomočkov, kot so kalkulator, ura, kodna tabela standarda Unicode, ipd.

- Poleg teh gumbov so na pultu tudi gumbi, ki predstavljajo vsa trenutno odprta okna. Na vsakem izmed teh gumbov je napisano ime aplikacije in dokumenta. Klik na tak gumb nas prestavi v okno z ustrezno aplikacijo in dokumentom.

- Zraven teh gumbov je ikona s slovensko zastavo, ki nam pove, da imamo privzet slovenski razpored tipk. S klikom na to ikono lahko nastavimo angleško/ameriško razporeditev.

- Poleg nastavitve tipkovnice je še nekaj ikon, s katerimi lahko vplivamo na glasnost pri nekaterih multimedijskih napravah in aplikacijah.

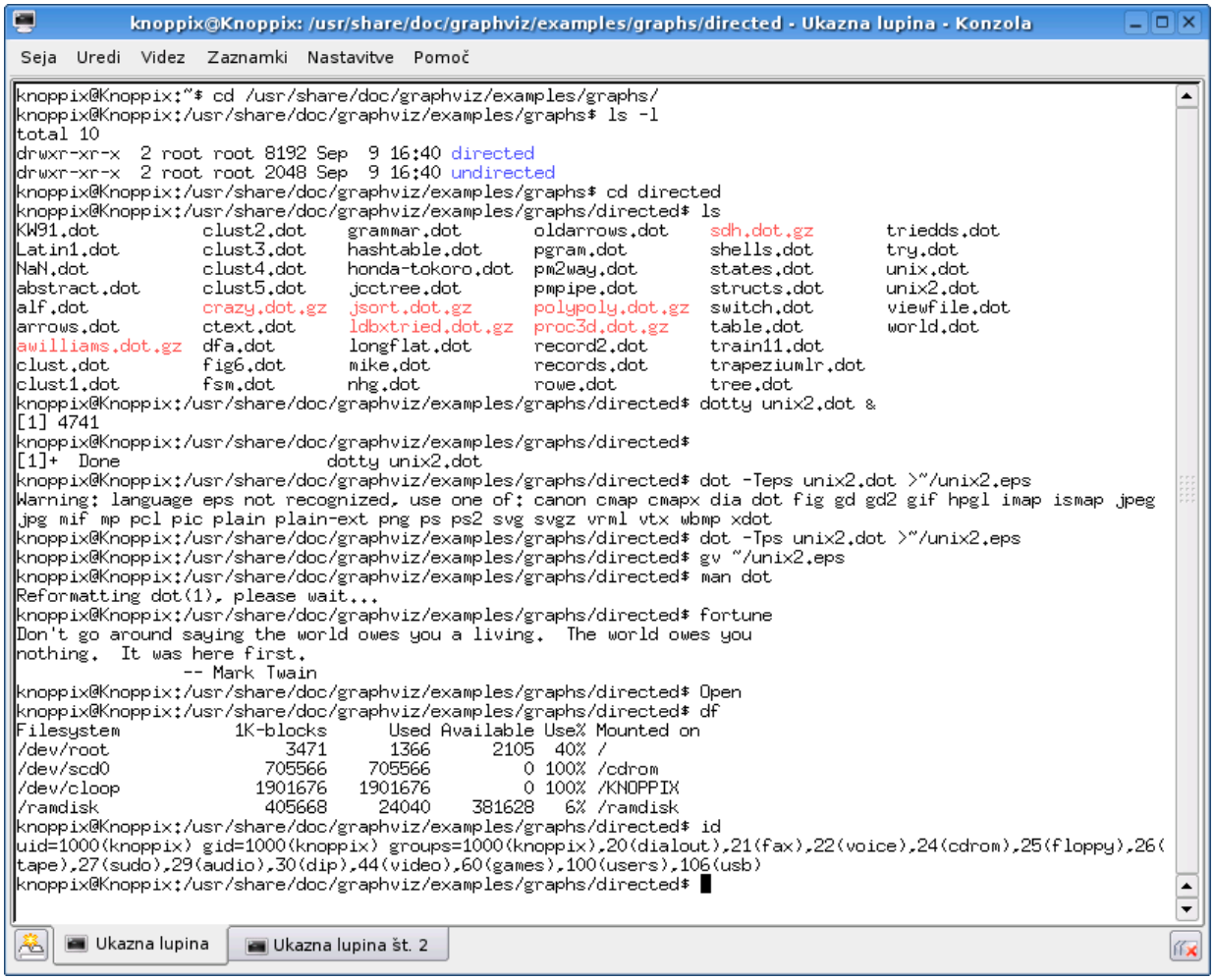
- V skrajno desnem kotu pulta pa se nahajajo še digitalna ura in gumbi za izbiro delovnih področij. Gumbi z oznakami “1”, “2”, “3” in “4” predstavljajo posamezna namizja, med katerimi lahko preklapljamo, tako da imamo v vsakem namizju manj odprtih oken in tako večji pregled na njimi. Število namizij lahko v nadzornem središču nastavimo vse do 20.

- Na samem namizju je nekaj ikon, s katerimi lahko dostopamo do aplikacij, vhodno/izhodnih naprav, omrežja, dokumentov in smeti, kamor odlagamo datoteke, ki jih ne potrebujemo več. Ikono na namizju aktiviramo z dvojnim klikom na miškin levi gumb, medtem ko za gumbe na pultu zadostuje enojni klik. Na namizje lahko sami dodajamo ikone, tako da z klikom na desni miškin gumb na namizju prikličemo meni in prek opcije “Ustvari novo” povemo, kakšno ikono želimo.

Če kazalec miške postavimo na določeno ikono, se nam izpiše kratek opis akcije, ki se izvrši, ko izberemo to ikono.

Kot rečeno pa je to le primer izgleda namizja, saj ga lahko vsak uporabnik prilagodi svojim potrebam!

Osnovne lastnosti oken



Slika 4.4: Primer terminalskega okna

Na sliki 4.4 vidimo primer terminalskega okna. Naslednje lastnosti veljajo za vsa okna[4]:

- V zgornjem desnem kotu okna imamo tri gumbe: s pritiskom na

[4]Te lastnosti veljajo po osnovnih nastavitvah, čeprav so odstopanja seveda možna na podlagi različnih distribucij operacijskega sistema (na primer Fedora, Mandrake itd) in različnih upravljalcev oken.

prvi gumb povzročimo, da je okno vidno na vseh delovnih področjih, drugi gumb okno zmanjša na velikost ikone na pultu, tretji pa okno poveča na celoten zaslon.

- Okno zapremo s pritiskom na gumb v levem zgornjem kotu okna.

- Po pritisku na naslovno vrstico okna z desnim miškinim gumbom se nam odpre meni, kjer so poleg omenjenih opcij dosegljive tudi sledeče: prestavljanje okna, spreminjanje velikosti okna, prestavitev okna na drugo delovno področje, skrivanje vsebine okna in spreminjanje izgleda okna. V primeru s slike 4.4 moramo torej pritisniti na vrstico z napisom *Shell - Konsole*.

- Vsebino okna lahko skrijemo tudi tako, da dvakrat kliknemo na naslovno vrstico okna z levim miškinim gumbom.

- Okno lahko premaknemo tudi tako, da z miško izberemo[5] naslovno vrstico okna in prestavimo okno. Ko je okno na želenem mestu, spustimo gumb miške.

- Tudi velikost okna lahko spremenimo z uporabo miške: postavimo se na rob okna tako, da znak za miškin kazalec spremeni svojo obliko, nato pa pritisnemo in držimo gumb miške in spremenimo velikost okna. Po končani operaciji sprostimo miškin gumb.

- V notranjosti okna je prikazana njegova vsebina, pa naj bo to besedilo, grafika ali kombinacija obeh.

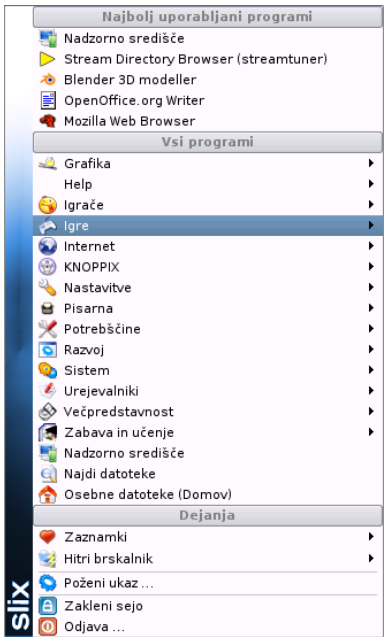
4.4.3 Osnovni meni – od pomoči do aplikacij

Sedaj pa si oglejmo opcije, ki jih dobimo na izbiro po pritisku na izhodiščni gumb. Meni, ki se nam odpre po pritisku, je prikazan na sliki 4.5. Vidimo lahko, da je opcij veliko, izpostavimo pa le nekaj osnovnih:

- Aplikacije so zbrane v več različnih podmenijih glede na lastnosti aplikacije. Tako imamo podmeni za grafične aplikacije “Graphics”, podmeni za internetne aplikacije “Internet”, podmeni za večpredstavne aplikacije “Multimedia”, podmeni za igre “Games” itd.

- Opcija “Find Files” je namenjena iskanju datotek.

[5]Za izbiro vedno uporabljamo miškin levi gumb. Pod Linuxom oziroma Unixom ima miška tipično tri gumbe. Za KDE je tudi značilno, da se akcija, ki se skriva pod neko ikono, zažene po prvem pritisku na ikono ali drugem zaporednem pritisku, kar lahko nastavimo v nadzorni plošči z opcijo “Peripherals→Mouse”.



Slika 4.5: Osnovni meni

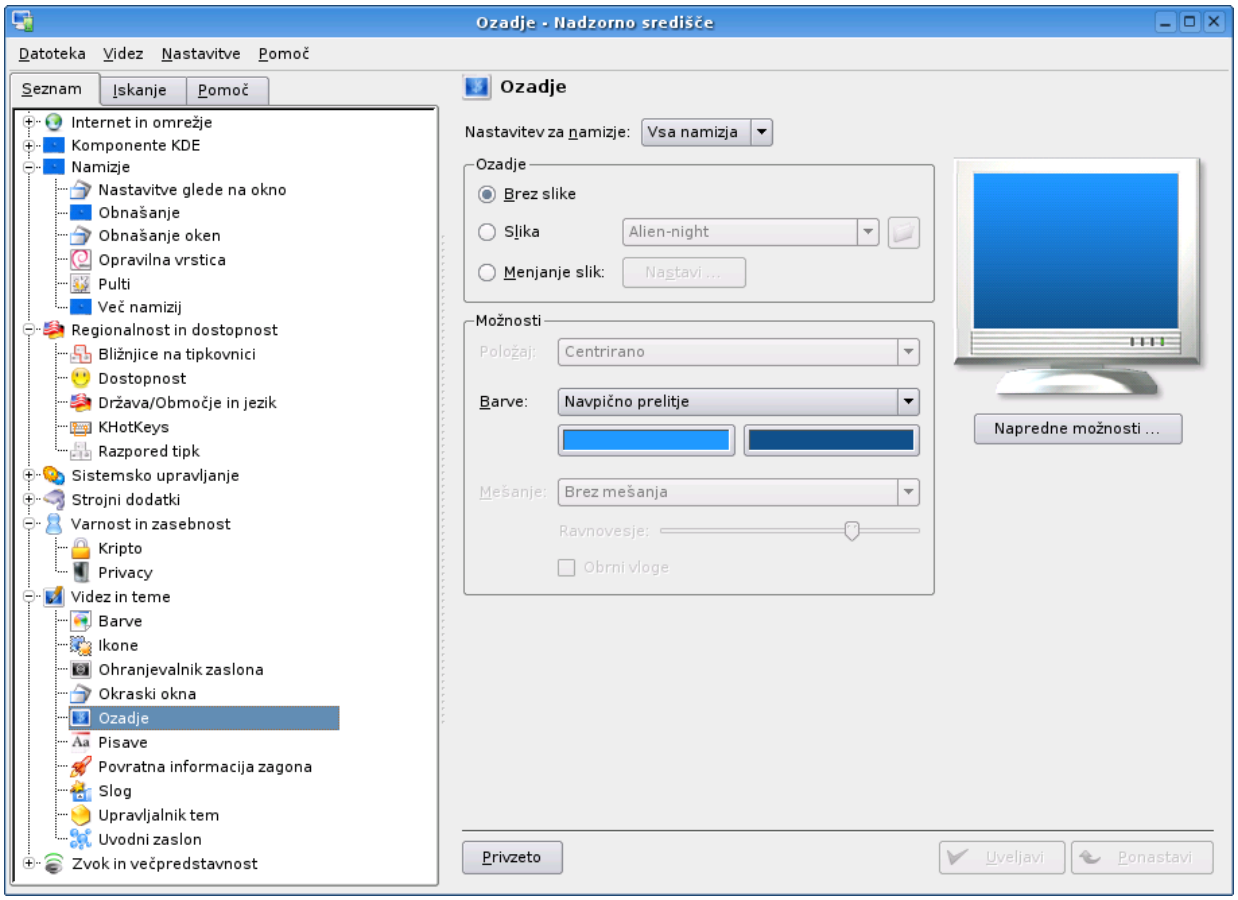
- Opcija "Personal Files (Home)" zažene brkljalnik, ki nam prikaže strukturo osnovnega imenika prijavljenega uporabnika in nam omogoča osnovno manipulacijo z datotekami.
- Opcija "Control Center" zažene kontrolno ploščo.
- Opcija "Help" zažene pomoč na temo Namizje KDE.
- Opcija "Lock Screen" zaklene računalnik pred nepovabljenimi gosti. Uporabnik lahko nadaljuje z delom takoj, ko vpiše svoje geslo in ga potrdi s pritiskom na gumb tipkovnice `<Enter>`.
- Opcija "Logout" odjavi trenutnega uporabnika. Po odjavi se prikaže prijavno okno. Če v prijavnem oknu izberemo opcijo "Shutdown", se računalnik pripravi na izklop. Računalnik izklopimo šele tedaj, ko nam le-ta javi, da je sistem v celoti zaustavljen[6].

4.4.4 Nadzorna plošča – spreminjanje videza in lastnosti namizja

V nadzorni plošči lahko vsak uporabnik nastavi videz in lastnosti namizja tako, kot si sam želi. Omenimo le nekaj možnosti:

- Če izberemo opcijo "Desktop→Panels", lahko nastavimo izgled in lastnosti opravilne vrstice, poimenujemo delovna področja ipd.

[6] Novejši računalniki se izklopijo sami.



Slika 4.6: Izgled nadzorne plošče v primeru, ko želimo spreminjati ozadje namizja

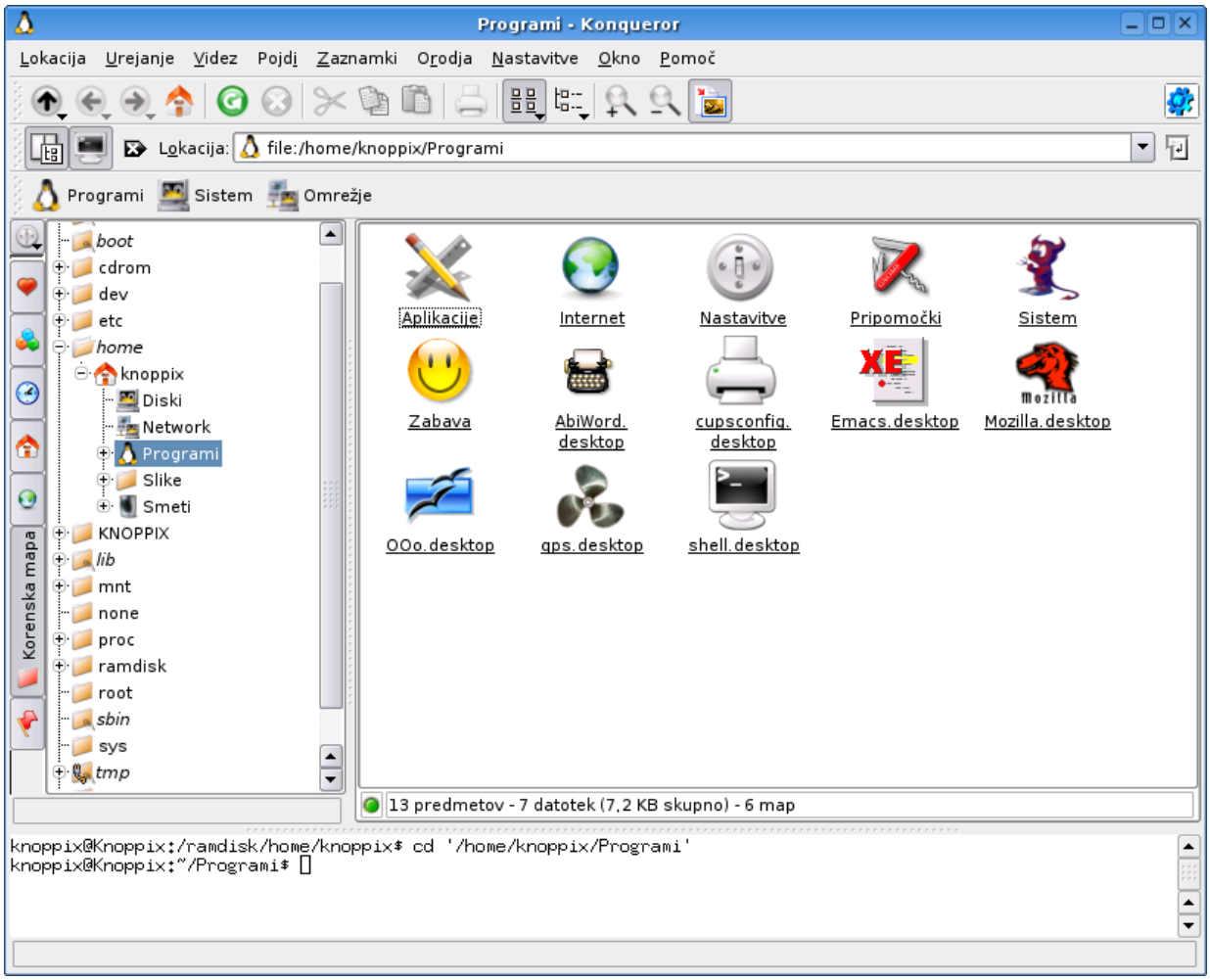
- Če izberemo opcijo “Appearance & Themes→Background”, lahko spreminjamo ozadje namizja. Kako izgleda kontrolna plošča v tem primeru, je razvidno s slike 4.6[7].
- Če izberemo opcijo “Appearance & Themes→Colors”, lahko spreminjamo barvni izgled gradnikov oken.
- Če izberemo opcijo “Appearance & Themes→Fonts”, lahko spreminjamo pisavo, ki jo uporabljamo za prikaz besedila na namizju.
- Če izberemo opcijo “Appearance & Themes→Screensaver”, lahko izberemo ohranjevalnik zaslona.
- Če izberemo opcijo “Desktop→Window Behavior”, lahko določimo akcije za različne situacije na namizju. Na primer, določimo lahko,

[7]Opcija “Appearance & Themes” se v predhodnih različicah namizja KDE imenuje tudi “Look & Feel”.

kaj se zgodi v primeru, ko pritisnemo na naslovno vrstico okna, če je okno aktivno: ali naj se maksimizira ali naj se pomanjša tako, da ostane le naslovna vrstica ipd.

4.4.5 Delo z datotekami

Delo z datotekami na ravni grafičnega uporabniškega vmesnika poteka po principu povleci in spusti (angl. *drag and drop*). To pomeni, da lahko praktično vse osnovne operacije izvedemo zgolj z uporabo miške.



Slika 4.7: Osnovno okno za manipulacijo z datotekami upravljalca datotek Konqueror

Preden pa si ogledamo osnovne operacije manipulacije z datotekami, navedimo nekaj dejstev o upravljalcu datotek (angl. *File Manager*), ki vse to omogoča. Upravljalec datotek pod namizjem KDE 3.3 se imenuje

Konqueror[8]. Osnovno okno za manipulacijo z datotekami tega programa prikazuje slika 4.7. Konqueror pa ni le upravljalec datotek, čeprav je to njegova osnovna naloga, pač pa v sebi skriva tudi spletni brskalnik[9], odjemalca FTP (File Transfer Protocol) za prenos datotek po mreži, hkrati pa služi kot strežnik za ostale aplikacije KDE, saj jim nudi svoje mrežne storitve in tako omogoča transparenten dostop do datotek.

Veliko upravljalcev datotek pod Unixom ne podpira namizja, kot smo ga vajeni danes. Namizje namreč ni drugega kot okno, v katerem lahko manipuliramo z datotekami. Konqueror pa je eden izmed upravljalcev, ki v tem pogledu pričara podobnost z operacijskimi sistemi, kot so OS/2, MacOS in Windows.

Ustvarjanje imenika

Pod Linuxom obstaja več načinov ustvarjanja imenika, mi pa si oglejmo naslednja dva:

- V osnovnem oknu za manipulacijo z datotekami upravljalca datotek Konqueror izberemo meni "Edit", podmeni "Create new" in znotraj njega opcijo "Folder". Ko imeniku podamo ime, se to prikaže na želenem oziroma trenutnem mestu v datotečni strukturi. Kot vsak element grafičnega uporabniškega vmesnika, je tudi imenik predstavljen z ustrezno ikono.
- Če na mestu, kjer želimo ustvariti imenik, pritisnemo na desni gumb miške, se nam prikaže priročni meni, ki med drugimi opcijami ponuja tudi podmeni "Create new". Nov imenik sedaj ustvarimo podobno kot zgoraj, saj moramo znotraj tega podmenija izbrati opcijo "Folder".

Ustvarjanje bližnjic, kopiranje in premikanje

Vse tri akcije izvršimo na zelo podoben način. Če želimo na primer nek imenik premakniti na namizje, ga tja skopirati ali na njemu ustvariti bližnjico do tega imenika, uporabimo princip vleci in spusti:

1. Imenik označimo z levim gumbom miške, vendar gumba ne spustimo.
2. Imenik potegnemo na želeno mesto in spustimo miškin gumb.
3. Na mestu, kjer smo gumb spustili, se nam pokaže meni, ki vsebuje štiri opcije: "Copy Here", "Move Here", "Link Here" in "Cancel".

[8]Upravljalec datotek pod namizjem KDE 1 se imenuje KFM.

[9]S tem je pojasnjena tudi velika podobnost v videzu in načinu delovanja spletnega brskalnika in upravljalca datotek Konqueror.

4. Če izberemo opcijo “Copy Here”, se nam bo imenik na želeno mesto skopiral; če izberemo opcijo “Move Here”, se nam bo imenik na želeno mesto premaknil; če pa izberemo opcijo “Link Here”, se nam bo na želenem mestu ustvarila bližnjica do tega imenika. Opcija “Cancel” operacijo prekliče.

Preimenovanje in spreminjanje dovoljenj

Če na primer nad ikono imenika kliknemo z desnim gumbom miške, se nam prikaže kontekstni meni. Znotraj tega menija je tudi opcija “Properties”. Če izberemo to opcijo, se nam odpre okno, znotraj katerega lahko spremenimo ime in dovoljenja imenika. Spremembe potrdimo s pritiskom na gumb “OK”.

Brisanje

Ikona koša z imenom *Trash*, ki se nahaja na namizju, predstavlja imenik, v katerega spravljamo vse nepotrebne datoteke. Najprej si oglejmo dva načina, kako na primer spravimo imenik v koš:

- Imenik označimo z levim gumbom miške, vendar gumba ne spustimo. Nato imenik potegnemo na ikono koša ter nad košem sprostimo levi gumb miške.
- Nad imenikom pritisnemo z desnim gumbom miške in prikaže se kontekstni meni, ki vsebuje tudi opcijo “Move to Trash”. Če izberemo to opcijo, se želeni imenik prestavi v koš.

Če izberemo ikono koša na namizju, se nam prikaže njegova vsebina. Tako lahko datoteke tudi vzamemo iz koša. Če pa nad ikono koša pritisnemo desni gumb miške, se nam prikaže kontekstni meni, ki vsebuje tudi opcijo “Empty Trash Bin”. Izbira te opcije povzroči nepreklicno brisanje vsebine koša z diska.

Datoteke pa lahko brišemo z diska tudi direktno, torej ne preko koša. Če na primer nad izbranim imenikom pritisnemo na desni gumb miške, se nam prikaže kontekstni meni, znotraj katerega je tudi opcija “Delete”. Po izbiri te opcije se izbrani imenik nepreklicno zbriše z diska.

4.4.6 Še nekaj koristnih napotkov za delo v Linuxu

V tem poglavju smo želeli osvetliti razliko med pojmi XWin, namizje, upravljalec oken in upravljalec datotek. Nato smo si ogledali značilnosti namizja KDE 3.3, vendar smo omenili le tiste elemente, ki so pomembni za prve korake dela z grafičnim uporabniškim vmesnikom pod operacijskim sistemom Linux. Za konec pa omenimo le še nekaj priročnih dejstev:

- Izbira datoteke oziroma imenika s klikom na levi gumb miške povzroči, da se prikaže vsebina imenika ali izvede izvedljiva datoteka ali prikaže vsebina dokumenta. Klik na desni gumb povzroči, da se prikaže priročen kontekstni meni z vsemi osnovnimi ukazi, ki jih lahko uporabimo nad označeno datoteko oziroma imenikom. Pritisk na srednji gumb miške nad imenikom pa znotraj osnovnega okna za manipulacijo z datotekami upravljalca datotek Konqueror povzroči, da se vsebina imenika prikaže v novem oknu.

- Kako delamo z več datotekami in/ali imeniki hkrati? Pritisnemo tipko `<Control>` na tipkovnici in s pritiskom na levi gumb miške izberemo vse želene datoteke in/ali imenike. Na koncu sprostimo tipko `<Control>`. Nadaljnje akcije so identične primeru, ko operiramo le z enim imenikom oziroma eno datoteko. Klik na neoznačeno ikono ali izven ikone razveljavi izbor.

- Kako lahko najlažje natisnemo datoteko tipa PS (PostScript)? Enostavno tako, da datoteko potegnemo na ikono tiskalnika (če ta ikona na namizju obstaja), seveda pa mora tiskalnik razumeti jezik PS. Na podoben način lahko tiskamo tudi datoteke tipa TEX, DVI, ASCII ipd; če imamo ustrezno nastavljen tiskalnik.

- Znotraj okna upravljalca datotek Konqueror lahko med drugim pregledujemo tudi strani o delovanju Linuxovih ukazov. V okno poleg napisa "Location" vpišite `man:/` in ključno besedo. Na primer, če želimo izvedeti vse o ukazu `telnet`, vnesemo `man:/telnet` in pritisnemo na tipko `<Enter>`. Strani lahko vsebujejo tudi hipertekstne povezave. Podobno lahko uporabljamo tudi spletni protokol `http` in protokol `ftp`. Če na primer v okno vpišemo `http://www.lrv.fri.uni-lj.si` in pritisnemo na tipko `<Enter>`, se nam prikaže spletna stran, ki se nahaja na spletnem naslovu `www.lrv.fri.uni-lj.si`. Če pa na primer vpišemo `ftp://lrv.fri.uni-lj.si`, se nam prikaže vsebina diska na zahtevani lokaciji. Seveda moramo v slednjem primeru imeti ustrezna dostopna dovoljenja. Če ta dovoljenja imamo, se nam vsebina oddaljene lokacije prikaže tako, kot da je del našega lokalnega diska. To pomeni, da lahko z datotekami na oddaljeni lokaciji manipuliramo enako, kot če bi bile del našega računalnika.

4.5 Naloge

1. V Linuxovi lupini ustvarite imenik `~/vaja`.

2. V Linuxovi lupini prekopirajte vse datoteke, katerih ime se začne s črko `c`, z diskete v imenik `~/vaja`.

3. V Linuxovi lupini poiščite vse datoteke, katerih ime se začne s črko `k` v imeniku `/usr/share` in vseh njegovih podimenikih.

4. V Linuxovi lupini vsem datotekam s končnico `tex` v trenutnem imeniku spremenite dovoljenja tako, da bodo vsi uporabniki imeli dovoljenje za branje teh datotek, pisanje in izvajanje pa bo dovoljeno samo lastniku.

5. V Linuxovi lupini dodelite lastništvo nad vsemi datotekami s končnico `mp3` v trenutnem imeniku in vseh podimenikih uporabniku z uporabniškim imenom `janez`.

6. V Linuxovi lupini ustvarite imenik `~/Dokumenti/slike`, nato pa vanj iz imenika `~/Dokumenti` prekopirajte vse slike, ki se začnejo na `p` (in imajo končnice `.png`).

7. V namizju KDE nastavite pisavo v naslovnih vrsticah oken na *Helvetica* z velikostjo 12.

8. Na površini namizja KDE ustvarite ikono, s pomočjo katere se bo ob kliku nanjo pognal program Emacs.

9. V sistemskem meniju namizja KDE poiščite program za peko zgoščenk (K3b) in z njim imenik ~ shranite na prazno zgoščenko.

10. Program Konqueror nastavite tako, da bo ob vsakem zagonu najprej naložil internetno stran `http://www.linux.org`.

4.6 Koristne spletne povezave

1. Namestitev in začetek dela z Linuxom:
   `http://www.redhat.com/mirrors/LDP/LDP/gs/gs.html` (angleško)
   `http://www.lugos.si/delo/slo/LIGS-sl/node1.html` (slovensko)

2. Za namestitev operacijskih sistemov Linux in Windows na istem računalniku:
   `http://www.redhat.com/mirrors/LDP/LDP/Linux+Windows-GUIDE/index.html`

3. Priročnik za uporabo Linuxa (angl. *Linux Users Guide*):
   `http://www.redhat.com/mirrors/LDP/docs.html#guide`

4. Spletna stran razvijalcev namizja KDE:
   `http://www.kde.org/`

5. Osnove upravljalcev oken (angl. *Window Managers for X - The Basics*):
   `http://www.plig.net/xwinman/basics.html`

6. Slovenščina in Linux:
   `http://nl.ijs.si/GNUsl/tex/tslovene/slolang/node26.html`
7. Skupina za slovenjenje Linuxa:
   `http://www.lugos.si/delo/slo/`
8. Slix – Slovenski živi Linux:
   `http://slix.ljudmila.org/`
9. Knoppix Linux:
   `http://www.knoppix.net/`
10. Spisek živih Linuxov:
    `http://www.frozentech.com/content/livecd.php`
11. DistroWatch.com: Put the fun back into computing – Use Linux:
    `http://distrowatch.com/`

4.7 Literatura

[1] S. Cvjetović, A. Košir, R. Maurer, P. Peterlin, and M. Samastur. *Linux z namizjem KDE*. Pasadena, Ljubljana, 2000.

[2] A. Košir, R. Maurer, P. Peterlin, and M. Samastur. *Namestimo Linux*. Pasadena, Ljubljana, 2001.

[3] M. Welsh and L. Kaufman. *Running Linux*. O'Reilly & Associates, USA, 1996.

5

Urejanje besedil

Ko so se konec 70-ih let pojavili tako imenovani "miniračunalniki", so se začeli razvijati tudi programi za računalniško urejanje besedil. Prednosti urejanja besedil z računalnikom so očitne. Z njim lahko hitreje in lažje spreminjamo, popravljamo in preoblikujemo besedila. Prvi računalniški programi za oblikovanje besedil so bili precej okorni in njihovi končni izdelki po videzu niso presegali besedil natipkanih s pisalnimi stroji. Z razvojem novih vrst tiskalnikov in sodobnih programskih orodij pa je računalniško oblikovanje besedil omogočilo tako imenovano namizno založništvo in hkrati povzročilo revolucijo v celotni tiskarski in založniški dejavnosti. Osebni računalniki so v začetku 90-ih let pri pisanju in urejanju besedil praktično povsem izrinili klasične pisalne stroje.

V tem poglavju se bomo omejili predvsem na tista programska orodja, ki so namenjena pisanju in oblikovanju besedil, ne pa na tista, ki so v osnovi namenjena nadaljnjemu oblikovanju že obstoječih besedil in njihovi kombinaciji s slikovnim gradivom (npr. program PageMaker). Ker so danes tako rekoč vsakemu uporabniku dostopna zelo zmogljiva orodja za oblikovanje besedil, je težko potegniti ločnico med običajnim oblikovanjem kratkega besedila, kot je na primer poslovno pismo, in zahtevnim tipografskim oblikovanjem, kot ga zahteva neka knjiga. Začetniki grešijo predvsem pri pretirani uporabi in nesmotrni kombinaciji raznovrstnih možnosti, ki jih ta orodja omogočajo. Na primer, z uporabo prevelikega števila različnih in med seboj neusklajenih vrst pisav v istem dokumentu. Zato je še toliko bolj pomembno, da se vsi uporabniki zavedamo osnovnega izhodišča pri oblikovanju vseh besedil: *namen dobre tipografije je lažje* ***razumevanje*** *vsebine besedila!*

5.1 Načini urejanja besedil

Računalniške programe za urejanje besedil ločimo glede na osnovno metodo urejanja na dve veliki skupini:

vizualno urejanje po načelu "kar vidiš, to dobiš" (angl. WYSIWYG – What You See Is What You Get) in

logično urejanje besedil.

5.1.1 Vizualno urejanje besedil

Osnovni princip vizualnega urejanja besedil je, da so rezultati vseh uporabnikovih akcij takoj vidni na zaslonu. Programi za vizualno urejanje besedil zahtevajo sicer sodobno strojno opremo in grafični uporabniški vmesnik, vendar pa so enostavni za uporabo in se jih je možno hitro naučiti, saj je delo z njimi konceptualno zelo podobno pisanju na pisalnem stroju. Hkrati ko pišemo besedilo, se odločamo tudi o njegovi obliki (poravnavi, vrsti in velikosti pisave). Možno je tudi enostavno vključevanje slikovnega gradiva.

Najbolj znani primeri programov za vizualno urejanje besedil so Wordstar, WordPerfect, Write in Word. Kot primer orodja za vizualno urejanje besedil bomo v tem poglavju obravnavali program OpenOffice.org Writer iz pisarniške zbirke OpenOffice.org.

5.1.2 Logično urejanje besedil

Logično urejanje besedil izhaja iz tradicije tiskarstva. Nekoč so besedilo oziroma rokopis, ki ga je bilo potrebno natisniti, uredniki oziroma tiskarji najprej opremili z navodili, kako naj bodo posamezni deli rokopisa natisnjeni. To pomeni, da so v rokopisu ročno označili velikost črk, razmike med posameznimi deli besedila, velikost črk za naslove in podobno. Ta navodila so nato upoštevali stavci pri stavljenju besedila. Večina založnikov pa je razvila hišne standarde o videzu določenih zvrsti besedil. Če je nek časopis na primer uporabljal pet različnih velikosti naslovov, uredniku ni bilo potrebno vsakokrat znova pisati vseh ukazov o poravnavi, vrsti, velikosti in teži pisave, temveč je naslov enostavno označil z vrsto naslova. Stavec pa je potem pogledal v navodila, kako se mora oblikovati izbrana vrsta naslova. Tako posredno določanje nekih lastnosti je izredno koristno načelo, ki se ga tudi sicer pogosto uporablja v računalništvu.

Ko so se začeli uporabljati računalniki za oblikovanje besedil, so prvi programi tudi zahtevali vsakokraten vnos tipografskih ukazov. ÂŽe konec 60ih let pa se je začelo uveljavljati posredno označevanje besedil

(angl. *markup*). Pod vodstvom Charlesa Goldfarba so v podjetju IBM razvili GML (Generalized Markup Language), ki je leta 1986 postal SGML (Standard Generalized Markup Language) oziroma standard ISO 8879 [8]. Da bi zadostil vsem možnim uporabnikom, je SGML moral postati izredno kompleksen, kar pa ni bilo združljivo z osebnimi računalniki, ki so se pojavili v istem obdobju. SGML je danes pravzaprav neke vrste metajezik, ki definira pravila za pisanje jezikov za označevanje besedil.

Pri delu s programi za logično urejanje besedil moramo osnovno besedilo opremiti z ustreznimi ukazi, ki določajo posamezne sestavine v besedilu (naslove, podnaslove, odstavke, sklice, enačbe itd.) ali pa neposredno oblikujejo besedilo (npr. vrsta in velikost pisave). Posebni prevajalnik nato to z ukazi opremljeno besedilo prevede v končno obliko. Ta način urejanja se imenuje logični zato, ker od uporabnika zahteva predvsem, da v besedilu označi njegove logične komponente. Tako logično označevanje delov besedila nam omogoča, da se nam med samim ustvarjanjem besedila ni potrebno ukvarjati z videzom besedila, saj videz posameznih logičnih enot besedila določi prevajalnik na enoten način s pomočjo posebnih slogovnih datotek. Datoteke, ki določajo videz, lahko seveda spremenimo in tako je možno konsistentno, glede na strukturo besedila, poljubno spremeniti obliko besedila. Tako je oblikovanje (videz) neke komponente besedila sicer ločeno od te komponente, vendar odvisno od funkcije oziroma mesta te komponente v besedilu. Z ločitvijo videza in osnovne logične strukture elektronsko besedilo tudi ni več ujeto v en sam statični videz.

Primeri programskih orodij za logično urejevanje besedil so: Scribe, Nroff, Troff, TEX, LATEX, HTML in XML. Nekatere preproste elemente logičnega urejanja s pomočjo stilskih datotek poznajo tudi nekateri vizualni urejevalniki besedil, na primer tako imenovano mehko oblikovanje v OpenOffice.org Writerju. V nadaljevanju poglavja bomo kot primer logičnega urejanja besedil spoznali sistem LATEX.

5.1.3 Pisave

Pisava je eden največjih človeških izumov, pri čemer so različne kulture razvile svoje lastne pisave. Velik del sodobnega sveta uporablja latinico za pisanje svojega jezika. Kot pove že samo ime, so latinsko pisavo izumili v starem Rimu na prehodu 7. v 6. stoletje pred našim štetjem [16]. Najprej se je razvilo več vrst velikih črk ali majuskul: rimska monumentalna kapitala, rustikalna kapitala in kvadratična kapitala. Zaradi večje hitrosti pisanja in različnih vrst pisal (dleto, čopič, trstika) se je v 3. stoletju iz velike pisave razvila nova mala pisava ali minuskula. Prva mala pisava je bila mlajša rimska kurziva, ki je postala osnova številnim srednjeveškim pisavam. Za razliko od velikih pisav, ki so bile oblikovane v dvolinijskem

sistemu, so minuskulne pisave oblikovane v štirilinijskem sistemu.

Z renesanso in iznajdbo tiska pa se je pričel razvoj črkovnih slogov ali pisav, ki jih uporabljamo še danes, ko imamo na voljo v digitalni obliki več kot 100.000 različnih vrst latinskih pisav. Pisave se imenujejo največkrat bodisi po svojem oblikovalcu, naročniku, kraju nastanka ali namenu. Pisave lahko delimo po različnih kriterijih.

Proporcionalne pisave in pisave z enako širino črk

V proporcionalni pisavi imajo pri isti velikosti različne črke različno širino, ki je odvisna od oblike črke. Obstajo pa tudi pisave (angl. monospaced fonts), ki imajo vse črke enako široke (slika 5.1). Take črke so se razvile zaradi tehnoloških omejitev pisalnih strojev, saj je bila sprva širina za vse črke enako odmerjena.

iiiiiiiiii iiiiiiiiii

mmmmmmmmmm mmmmmmmmmm

Slika 5.1: Primerjava črk v proporcionalni pisavi in črk, ki so enako široke. Število i-jev in m-jev je povsod enako.

Če črke zavzemajo enako širino ne glede na svojo obliko, je med tankimi črkami (npr. i) več praznega prostora. Zaradi tega praznega prostora pa je bralcu težje razbrati razmejitve med posameznimi besedami, kar naredi tako besedilo v primerjavi s proporcionalno pisavo težje berljivo. Kljub temu pa so pisave s črkami enakih širin koristne za prikaz podatkov v tabelah zaradi navpične poravnave posameznih znakov, predvsem pa za prikaz programske kode, da se že na pogled loči od ostalega besedila. Črke enakih širin so tudi bolj primerne za računalniško urejanje besedil s pomočjo miške, saj je v proporcionalni pisavi težko z miško zadeti najožje črke kot so i, j ali t.

Dokumenti napisani z “monospaced” pisavo spominjajo na dokumente napisane s pisalnimi stroji, saj so ti zaradi mehanike morali imeti enako široke črke. Tako napisani dokumenti (pa še brez poravnanega desnega roba) imajo zato nekako bolj oseben in neformalen videz.

Serifne in tehnične pisave

Latinico delimo na dva glavna črkovna sklopa: sklop pisav s tankimi in podebeljenimi potezami (angl. *serif fonts*) ter sklop pisav s skoraj enako ali z enako debelimi črtami (tehnični slog – angl. *sans serif fonts*) [16].

Izvorno so serifi nastali pri klesanju velikih rimskih črk v kamen.
Neserifne ali moderne pisave pa so se razvile šele veliko kasneje (slika 5.2).

Slika 5.2: Primerjava pisave s serifi in brez serifov.

Oba sklopa pisav se delita še naprej v posamezne črkovne sloge, ki se imenujejo po zgodovinskih obdobjih, v katerih so pisave nastale [16].

Prvi **sklop pisav s serifi** delima na štiri skupine:

I skupina: beneške renesančne pisave (od 1470 do 1500),

II skupina: francoske renesančne pisave (od 1495 do 1757),

III skupina: baročne pisave (od 1757 do 1790),

IV skupina: klasicistične pisave (od 1790 od 1900).

Kasneje v 20. stoletju so se uporabljale vse skupine pisav s serifi in njihova uporaba ni bila omejena le na tisto časovno obdobje, v katerem so pisave prvotno nastale. To velja v še večji meri tudi danes, ko se s pomočjo nove računalniške tehnologije porajajo nove pisave ali se ponovno obujajo pisave, ki imajo značilnosti pisav izpred nekaj stoletij.

Drugi **sklop pisav tehničnega sloga**, kjer so črke povsod enako ali skoraj enako debele, delimo na dve skupini:

V skupina: egipčanske pisave oziroma pisave z oglatimi serifi (angl. *square serif types*) (od 1815 dalje). `V teh pisavah so serifi pravokotne črte`.

VI skupina: linearne pisave (od 1819 dalje) (angl. *sans serif types*). Te pisave nimajo serifov.

Obstajajo še štiri samostojne skupine črkovnih slogov:

VII skupina: dekorativne oziroma akcidenčne pisave. To so pisave, kjer so črke posebne oblike: votle, črtane, okrašene, osenčene, podvojene itd. Običajno so v takih pisavah le velike črke. Uporabljamo jih za vabila, poročna naznanila, vstopnice, oglase, plakate, naslove ipd.

VIII skupina: rokopisne pisave, ki so posnete po klasičnih kaligrafskih (lepopisnih) pisavah. Uporabljamo jih le za posamezne vrstice besedila ali za naslove. Mešanje z drugimi pisavami, če niso oblikovno posebej usklajene, ni priporočljivo.

IX skupina: risane pisave, ki so posnete po pisavah, napisanih s čopičem ali trsko. Pravila uporabe so podobna kot za rokopisne pisave.

V angleščini se skupine VII, VIII in IX skupaj imenujejo *decorative scripts*. Primer: *KALIGRAFSKA PISAVA*.

X skupina: gotske pisave. Johannes Gutenberg, izumitelj tiska, je na svojih prvih premičnih črkah upodobil pisavo tekstura. Gotske pisave so nastale zato, ker so prvi tiskarji želeli, da bi bile tiskane knjige čimbolj podobne rokopisnim pisavam tedanjega časa. Med gotske pisave štejemo: teksturo, rotundo, schwabacher in frakturo. V Nemčiji so najdlje uporabljali gotske pisave. Prvi dve slovenski knjigi – *Catechismus in der Windischen Sprach* in *Abecedarium* – obe iz leta 1550, sta bili natisnjeni z gotskimi pisavami. Trubar je uvidel, da so za slovenski jezik primernejše renesančne pisave in zato je bila tretja slovenska knjiga – *Abecedarium* – iz leta 1555 v celoti natisnjena v renesančni pisavi.

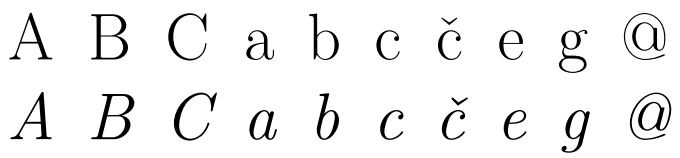
V matematiki posamezne gotske črke uporabljamo za označevanje simbolov. Tako na primer črki $\Re$ in $\Im$ uporabljamo za označevanje množice realnih in imaginarnih števil.

Izkušnje in raziskave o hitrosti in berljivosti natisnjenih besedil kažejo, da je pisave s serifi možno hitreje in lažje brati. Zato za daljša besedila običajno uporabljamo serifne pisave, za bolj poudarjene dele besedil, kot so naslovi in mednaslovi, pa se dobro obnesejo neserifne pisave.

Družine pisav

Črke ene pisave so navadno izdelane v več različicah, ki sestavljajo družino te pisave. Veliko pisav je združenih v družine pisav. Družino pisav sestavljajo pisave, ki so med seboj oblikovno usklajene, ločijo pa se po velikosti, teži, širini in obliki. Ni točnih pravil, katere različice morajo sestavljati neko družino pisav. V družini pisav so običajno pokončna pisava (angl. *roman*), *kurzivna pisava* (angl. *italic*), **krepka pisava** (angl. *bold*) in MALE KAPITELKE (angl. *small caps*) v nekaj standardnih velikostih.

V kurzivni pisavi so črke nagnjene v desno in v pravi kurzivni pisavi je oblika posameznih črk običajno tudi nekoliko drugačna od pokončnih črk. Primerjaj na primer črki *a* in *g* na sliki 5.3. V tako imenovani elektronski kurzivi pa so običajne pokončne črke *le nagnjene v desno*.



Slika 5.3: Primerjava pokončne in kurzivne pisave

Velikost pisav

Stopnje oziroma velikost črk so poimenovali že prvi tiskarji v drugi polovici 15. stoletja. V tiskarstvu ste se v 19. stoletju izoblikovala dva merska sistema: evropski ali Didotov tipografski merski sistem in anglosaški merski sistem. V dobi računalništva pa se je uveljavila tipografksa enota pika (angl. *point*), ki meri 1/72 palca ali 0,352778mm in jo je vpeljal Adobov jezik Postscript [1].

Tipične velikosti črk so 7, 8, 9, 10, 11, 12, 14, 16, 18, 20, 24, 30 in 36 pik. V klasični dobi tiskarstva so seveda morali imeti za vsako velikost poseben nabor svinčenih črk. V računalniški dobi pa lahko elektronsko povečamo ali pomanjšamo črke. Vendar poljubno pomanjšanje ali povečanje črk, ki so bile načrtovane za določeno ciljno velikost, ne daje dobrih rezultatov, saj se podoba črk za človeško oko ne spreminja linearno. Črke, načrtovane za večjo velikost, so običajno ožje in imajo več podrobnosti kot črke, ki so bile načrtovane za manjšo velikost, katerih poteze so širše (slika 5.4).

Pisava velikosti 12 pik in pisava velikosti 6 pik, povečana na velikost 12 pik.

Slika 5.4: Primerjava med črkami iste pisave, načrtovanimi za različno velikost

Poleg ogromne izbire različnih vrst pisav sodobna računalniška orodja za oblikovanje besedil omogočajo razne vrste manipulacij s pisavami, kot so senčene pisave ali pisave z obrisi, kjer je notranjost črk prazna. Tovrstne manipulacije so primerne le za grafično oblikovanje raznih logotipov, naslovnic itd., le redko pa pridejo v poštev za oblikovanje običajnih besedil.

Rastrske črke in obrisne črke

Črke prvih pisav, ki so se začele uporabljati na računalnikih, so bile sestavljene iz pik, saj je tako narekovala tehnologija (rastrski zasloni in matrični tiskalniki). Prve izhodne naprave (zasloni in tiskalniki) so imele relativno majhno ločljivost in zato se je že s prostim očesom videlo, da so črke sestavljene iz posameznih pik. Z boljšimi napravami se je ločljivost sicer povečala, toda če tako imenovano rastrsko črko dovolj povečamo, postane raster ali njihova mrežna struktura vidna prostemu očesu in zato

je njihov obris nazobčan.

Zato se je pojavil nov način definiranja oblik črk, in sicer z matematičnim zapisom krivulj, s katerimi so definirani obrisi črk (angl. outline fonts). Prvi in najbolj razširjen standard oziroma jezik, ki omogoča tak zapis črk, je PostScript podjetja Adobe [1]. Kljub temu, da je možno obrisno zapisane črke poljubno povečati, ne da bi postale nazobčane, moramo pri izbiri pisave kljub temu upoštevati njihovo "naravno" velikost (slika 5.4). Najnovejši standard obrisnih pisav je OpenType, ki je odpravil nekompatibilnost formatov PostScript in TrueType.

5.1.4 Osnovna izhodišča za oblikovanje besedil

Sodobna orodja za urejanje in oblikovanje besedil začetnika kaj hitro zmedejo zaradi velikega števila možnih vrst pisav ter drugih nastavitev (razmik med vrsticami, širina robov, napisi v glavi in vznožju strani itd.). Ena najpogostejših začetniških napak pri oblikovanju besedil je, da v enem dokumentu uporabljamo preveč različnih vrst pisav.

Osnovno izhodišče pri oblikovanju vsakega besedila je, da moramo *dokument oblikovati glede na njegov namen in funkcijo*! Zato je najbolj zanesljivo, da uporabimo standardne vzorce za osnovne vrste dokumentov, ki so pogosto na voljo v posameznih programskih orodjih. Sicer pa se zgledujmo po dobro oblikovanih in strukturiranih dokumentih [6]. Uporabljajmo standardne nastavitve širine robov, razmika med vrsticami, razmika med črkami (spiranje ali angl. *kerning*). Glede velikosti črk in širine vrstic moramo paziti, da bo v vrsticah deset do dvanajst besed oziroma največ okoli 60 znakov.

Med študijem se študenti srečajo predvsem s pisanjem poročil, seminarskih in diplomskih nalog. Več o pisanju strokovnih besedil si lahko bralec prebere v [6, 15].

Seminarske in diplomske naloge

Standardna struktura seminarskih in diplomskih nalog, pa tudi drugih strokovnih in znanstvenih besedil, je naslednja:

1. **Naslov** dela, imena avtorjev, institucija in njen naslov, naslovi elektronske pošte avtorjev, datum in kraj nastanka dela.

2. **Povzetek**, v katerem v sto do dvesto besedah opišemo vsebino celotnega dela (v slovenščini in tujem jeziku).

3. **Uvod**, s katerim uvedemo bralca v delo in podamo pregled celotnega dela.

4. **Pregled** področja, s katerim se delo ukvarja, in sorodnih rešitev, če predlagamo v delu neko novo rešitev. Če gre za kratko besedilo, sta uvod in pregled lahko združena.

5. **Glavni del**, ki je običajno sestavljen iz večih poglavij (npr. zajem podatkov, uporabljene metode in tehnologija, opis lastnih rešitev, rezultati).

6. **Zaključek**, kjer povzamemo glavne rezultate naloge.

7. **Zahvala**, v kateri navedemo imena vseh, ki so nam pri delu pomagali, oziroma so delo omogočili. To so lahko fizične osebe ali tudi institucije, ki so financirale naše delo.

8. **Seznam** virov oziroma literature, na katero se sklicujemo v svojem delu.

9. **Dodatki** (npr. algoritmi, daljše matematične izpeljave itd.).

Posebej moramo paziti, da se v samem besedilu na ustreznem mestu sklicujemo na uporabljene vire. Povsod, kjer lahko dosežemo boljšo razumljivost, uporabimo primerno slikovno gradivo (diagrame, grafe, tabele). Paziti moramo na primeren in razumljiv jezik ter ustrezno terminologijo.

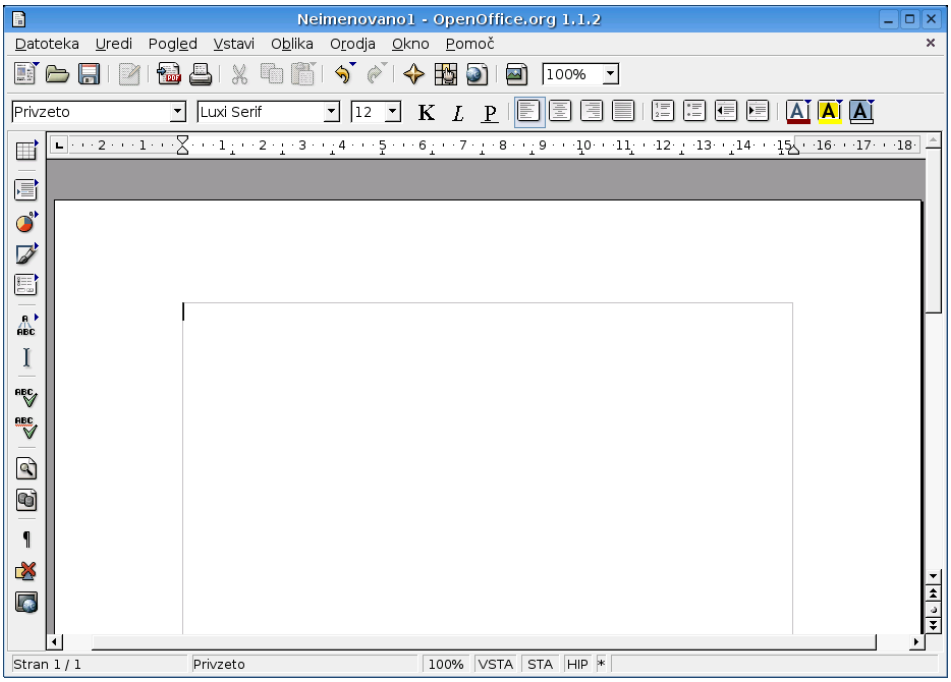
Večina orodij za urejanje besedil je bilo razvitih v ZDA in so zato prilagojena angleški rabi. Tako je vgrajena pomoč za ameriški pravopis, pravila za deljenje angleških besed, tudi pisave (npr. ligature – določene kombinacije črk, ki so oblikovane kot enota) so prilagojene angleški rabi. Paziti je potrebno tudi na pravilno rabo drugih tipografskih elementov, na primer narekovajev in pomišljajev, ki so različni v angleščini in slovenščini. Nekatera orodja podpirajo še nekatere druge svetovne jezike; francoščino, nemščino, španščino itd., skoraj noben program pa ne podpira v enaki meri tudi slovenščine. Edino LaTeX zaradi svoje prilagodljivosti kot neke vrste programski jezik zelo dobro podpira rabo slovenskega jezika kot tudi manjših jezikov (paket Babel). V sistemu LaTeX je možno uporabiti tudi slovenska pravila za deljenje besed.

5.2 OpenOffice.org Writer

OpenOffice.org Writer je program za vizualno urejevanje besedil. Vsebuje širok nabor funkcij, s katerimi lahko na enostaven način ustvarimo privlačne dokumente z besedilom, opremljenim s slikovnimi in drugimi elementi.

5.2.1 Vnos besedila

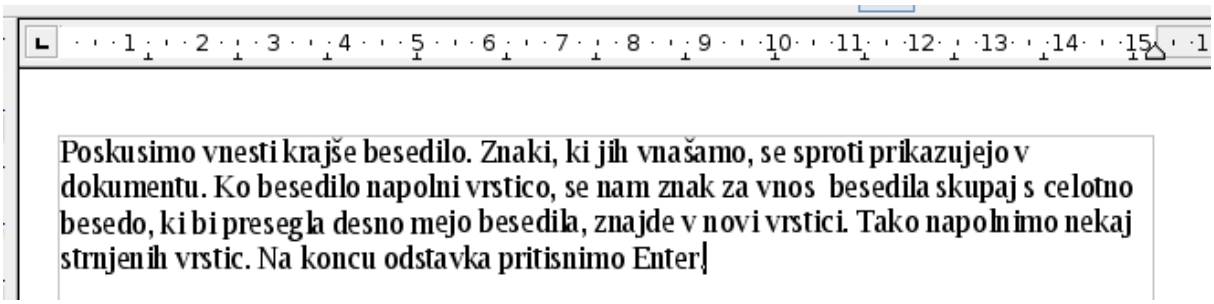
Ob zagonu programa ali ko ustvarimo nov dokument, se znajdemo pred oknom, podobnim tistemu na sliki 5.2.1 (takoj po namestitvi lahko nad dokumentom plava okno s slogovnikom, slika 5.16, ki pa ga skrijemo s klikom na križec v njegovem desnem zgornjem kotu).



Slika 5.5: Okno z novim dokumentom

V osrednjem delu okna programa vidimo vsebino dokumenta. Ker z Writerjem urejamo besedilo, ima dokument videz strani, enako pa bo dokument zgledal v svoji končni obliki, ko ga bomo natisnili na papir. V oknu vidimo le del dokumenta, z vodoravnim in navpičnim drsnikom pa lahko izberemo, kateri del dokumenta vidimo v oknu. Siv pravokotnik označuje robove besedila znotraj posamezne strani. Neposredno nad dokumentom vidimo ravnilo, ki označuje levo in desno mejo besedila. Utripajoča navpična črta znotraj strani označuje kazalko, tj. mesto, kamor bo program vstavil natipkano besedilo.

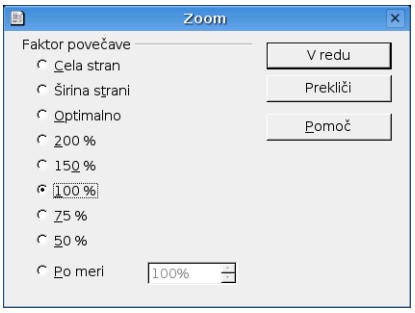
Poskusimo vnesti krajše besedilo. Znaki, ki jih vnašamo, se sproti prikazujejo v dokumentu. Ko besedilo napolni vrstico, se kazalka skupaj s celotno besedo, ki bi presegla desno mejo besedila, znajde v novi vrstici. Tako napolnimo nekaj strnjenih vrstic, ki tvorijo odstavek. Na koncu odstavka pritisnemo `<Enter>`, ki prenese kazalko na začetek novega odstavka. slika 5.6 prikazuje primer rezultata kratkega vnosa v dokument.



Slika 5.6: Videz okna z dokumentom po vnosu krajšega besedila

5.2.2 Pregledovanje dokumenta in premikanje po njem

Med urejanjem in pregledovanjem se po dokumentu premikamo tako, da spreminjamo mesto, ki ga trenutno pregledujemo, ali premikamo kazalko.



Slika 5.7: Določanje povečave pogleda na dokument

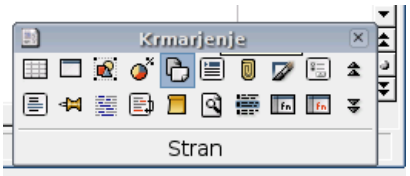
Obseg dokumenta, ki ga program lahko prikaže v svojem oknu, je odvisen od velikosti okna, mer dokumenta ter trenutne povečave. S povečavo (*zoom*) lahko povečamo pogled na dokument ali ga zmanjšamo do posamezne strani ali skupka strani. Povečavo nastavljamo s pogovornim oknu, ki ga prikličemo s "Pogled→Zoom ..." (slika 5.2.2). V pogovornem oknom lahko nato izbiramo med faktorji povečave, pri čemer prvi trije določijo prilagodljivo povečavo, ki je odvisna od velikosti okna in velikosti dokumenta, preostali faktorji pa so fiksni:

- **Cela stran**: prikaz celotne strani, ki jo trenutno pregledujemo ali urejamo.
- **ÂŠirina strani**: v oknu vidimo dokument od levega do desnega roba strani, ki jo trenutni pregledujemo.
- **Optimalno**: faktor povečave, ki omogoča pogled širine besedila od leve do desne meje.

- **200%, 150%, 100%, 75%, 50%**: fiksne vrednosti faktorja povečave.
- **Po meri**: fiksna vrednost povečave, ki jo sami vnesemo v polje.

Če imamo miško s koleščkom namesto srednjega gumba, lahko spreminjamo povečavo tako, da držimo tipko <Ctrl> in hkrati pomikamo kolešček navzgor za večjo povečavo ali navzdol za manjšo povečavo.

Faktor povečave prikazuje vrstica stanja. Če na vrednost dvokliknemo, prikličemo pogovorno okno za nastavljanje zooma. Zoom lahko nastavljamo tudi preko padajočega menija v funkcijski vrstici.



Slika 5.8: Izbira elementov dokumenta, med katerimi lahko krmarimo

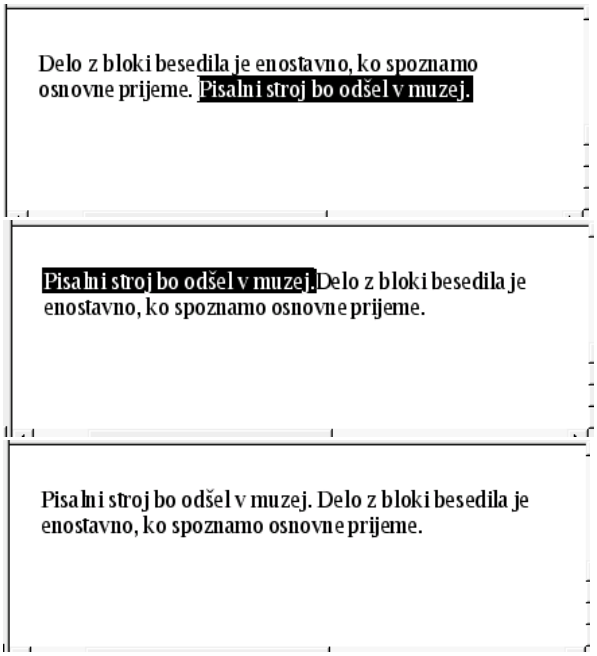
Drsniki določajo mesto pogleda dokumenta relativno glede na obseg dokumenta, ki ga prikazuje trenutno okno. Če se želimo hitro premikati med stranmi dokumenta, pa uporabimo tipki pod navpičnim drsnikom: za skok na predhodno stran in za skok na naslednjo stran. S pritiskom na tipko prikličemo plavajočo orodjarno "Krmarjenje" (slika 5.8), kjer izbiramo, med katerimi elementi dokumenta lahko krmarimo. Naziv izbrane vrste lahko preberemo na dnu orodjarne. Privzeta vrsta elementov za krmarjenje je stran, če pa izberemo katerikoli drug element, pa postanejo puščice na tipkah in modre barve.

Mesto, kamor vnašamo besedilo, lahko premikamo s smernimi tipkami na tipkovnici ali s klikom miške na želeno mesto. Vendar pa nam program pusti novo besedilo vstavljati le na tista mesta, kjer je že obstoječe besedilo, ali na konec (tudi praznega) odstavka. Poleg smernih tipk ima večina tipkovnic še tipke <Home> in <End> ter <Page Up> in <Page Down>. Vsako od teh tipk lahko uporabljamo samostojno ali v kombinaciji s tipko <Ctrl>, pri čemer slednja poveča korak premika:

- <←>, <→>: pomik za en znak levo ali v desno,
- <↑>, <↓>: pomik v prejšnjo ali naslednjo vrstico,
- <Ctrl>+<←>, <Ctrl>+<→>: premik za eno besedo v levo ali v desno,
- <Page Up>, <Page Down>: premik po dokumentu gor ali dol v dolžini v oknu vidnega dokumenta,

- <Home>, <End>: skok na začetek ali na konec trenutne vrstice,
- <Ctrl>+<Home>, <Ctrl>+<End>: skok na začetek ali na konec dokumenta.

5.2.3 Urejanje besedila



Slika 5.9: Označevanje in premikanje bloka besedila

Prednost dela z elektronskim dokumentom z besedilom je, da je nelinearno, saj lahko besedilo vnašamo v poljubnem vrstem redu, vrstni red spreminjamo, vstavljamo novo besedilo na poljubno mesto v besedilu ter brišemo zajetne dele besedila.

Pomen kazalke kot mesta za vnos besedila smo spoznali že v predhodnih razdelkih. Poleg vnosa novih znakov lahko znake tudi brišemo. Vračalka <Backspace> tako pobriše znak pred kazalko, tipka za brisanje <Delete> pa znak za kazalko. Poleg tega lahko znake, ki so za kazalko, prepišemo z novimi znaki, če vključimo prepisovalni način s pritiskom na tipko <Insert>. V način vstavljanja se vrnemo s ponovnim pritiskom na tipko <Insert>. Trenutno izbrani način ugotovimo iz vrstice stanja: oznaka VSTA nakazuje način, pri katerem nove znake vrivamo v besedilo, oznaka PREP pa pove, da bomo nove znake vnesli čez obstoječe znake.

Besedilo in ostale predmete, ki smo jih vnesli v dokument, lahko po dokumentu prestavljamo, jih množimo ali brišemo (slika 5.9). Ponovno vnašanje že obstoječega besedila je tako nepotrebno in nesmotrno, kar

velja za besedilo, ki ga hranimo v drugem dokumentu ali celo v drugih programih. Za te in druge operacije bomo spoznali delo z bloki besedila.

Izbran blok besedila je skupek znakov in drugih elementov, ki jih v vmesniku programa izberemo z miško, s tipkovnico ali z drugim ukazom. Najpreprostejši način izbiranja dela besedila je, da se z miško postavimo na začetek dela, ki ga želimo izbrati. Nato pritisnemo in držimo levo tipko miške ter miško prestavimo na konec dela besedila, ki ga želimo izbrati. ÂŽe med premikanjem opazimo, da program sproti prikazuje označeno besedilo z inverznimi barvami. Ko sprostimo tipko miške, je izbiranje besedila končano. Besedilo ostane izbrano, dokler ne kliknemo v okno z besedilo ali prestavimo kazalko.

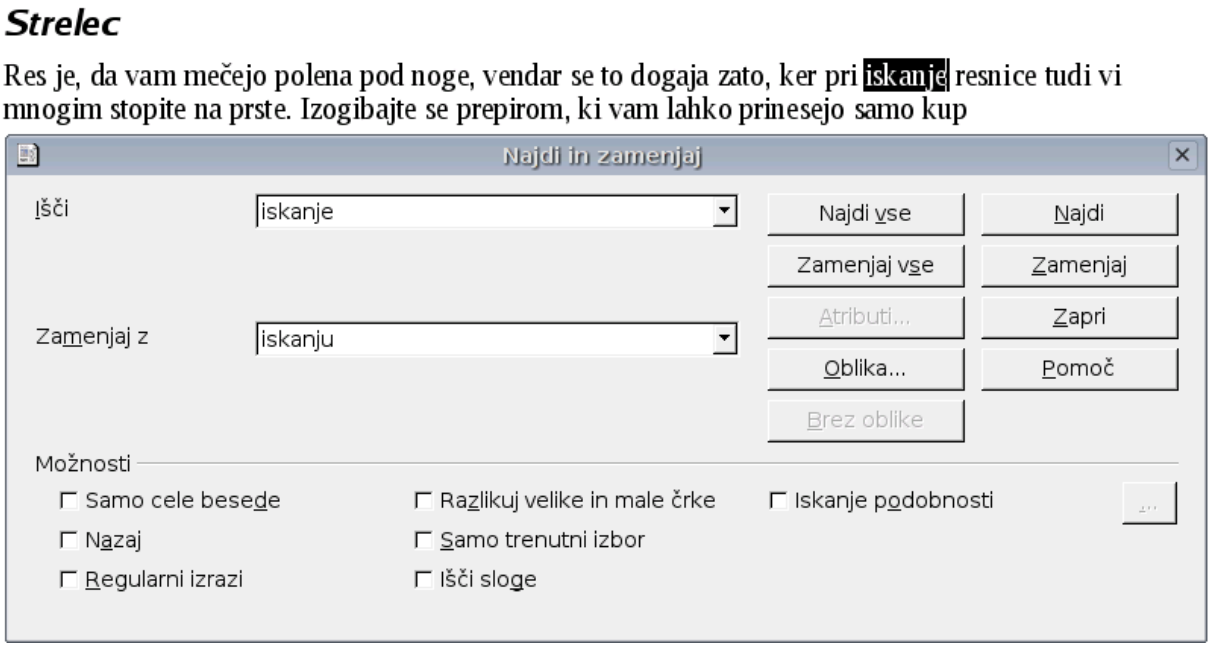
Tipka `<Shift>` v kombinaciji s premiki kazalke (tako z miško kot s smernimi in navigacijskimi tipkami na tipkovnici) omogoči širjenje ali krčenja izbora besedila. Na primer, če držimo tipko `<Shift>` in pritisnemo `<Control>+<→>`, bomo k izboru dodali besedo na desni strani kazalke.

Z miško lahko označimo besedo tako, da nanjo kliknemo dvakrat. Če v hitrem zaporedju na isto mesto kliknemo trikrat, bomo označili celotno vrstico. Če želimo izbrati celoten dokument, izberemo "Uredi→Izberi vse".

Na izbranem besedilu se bodo odrazile spremembe lastnosti znakov, odstavkov ipd. Izbrano besedilo lahko zbrišemo s pritiskom na tipko `<Delete>` ali `<Backspace>`, pa tudi katerakoli druga tipka bo povzročila, da izbrano besedilo nadomestimo z novim vnosom. Če se izkaže, da nismo želeli izbrisati toliko besedila, lahko iz menija "Uredi" ali iz funkcijske vrstice izberemo možnost "Razveljavi".

Izbrano besedilo prestavimo na drugo mesto v dokumentu tako, da postavimo miško nekam znotraj označenega besedila, nato pritisnemo (ne kliknemo!) levo miškino tipko ter premaknemo miško na mesto vstavljanja besedila. Kazalec miške nam bo nakazal, kaj se bo zgodilo, če na tistem mestu tipko miške izpustimo: znotraj izbora ne bo sprememb, izven njega pa bo program na ciljno mesto prenesel izbrano besedilo. Če med vlečenjem pritisnemo in držimo tipko `<Control>`, bomo na ciljnem mestu ustvarili *dvojnik* izbranega besedila. Opisani postopek deluje tudi, če miško prestavimo v okno drugega dokumenta z besedilom ali celo v okno drugega programa. Običajno je potem privzeta akcija ustvarjanje dvojnika izbranega besedila, če pa želimo besedilo prestaviti (tj. izbrisati iz izvirnega besedila), pa med vlečenjem držimo tipko `<Shift>`.

Nekoliko manj intuitivno, vendar zelo uporabno pri delu z bloki besedila je *odložišče* (angl. *clipboard*). Odložišče je del programa ali operacijskega sistema, ki začasno shrani del dokumenta. Do odložišča dostopamo posredno z uporabo ukazov "Izreži", "Kopiraj" in "Prilepi", ki jih najdemo tako v meniku "Uredi" kot na funkcijski vrstici. "Kopiraj" shrani izbrani del dokumenta na odložišče (nova vsebina odložišča



Slika 5.10: Pogovorno okno za iskanje in zamenjavo

običajno nadomesti predhodno shranjeno). "Izreži" je soroden ukazu "Kopiraj", le da izbrani del dokumenta izbriše. "Prilepi" pa vstavi dvojnik vsebine iz odložišča v dokument.

5.2.4 Iskanje in zamenjevanje

Iskanje mesta iskalnega niza ter zamenjavo nizov z drugim nizom omogočajo že osnovni urejevalniki besedil. OpenOffice.org Writer omogoča izboljšanje iskanja tako, da nastavljamo natančnost ujemanja z vzorcem in način primerjave.

Enostaven primer iskanja je iskanje besede. Denimo, da iščemo besedo `iskanje` (slika 5.10). Iskanje aktiviramo z "Uredi→Najdi in zamenjaj" ali s kombinacijo tipk `<Control>+<F>`. V polje "Išči" vnesemo niz, ki ga želimo poiskati v dokumentu, v našem primeru torej `iskanje`. Če po besedilu samo iščemo, pustimo polje "Zamenjaj z" prazno (oz. ne spreminjamo njegove vsebine), sicer pa ga izpolnimo z nizom, za katerega želimo, da bo v dokumentu nadomestil najdene nize iz polja "Išči". Vnesimo na primer niz `iskanju`.

Samo iskanje aktiviramo s klikom na "Najdi", zamenjavo pa s klikom na "Zamenjaj". V obeh primerih program poišče prvo pojavitev niza v polju "Išči". Ob vsakem naslednjem kliku tipke "Najdi" program poišče naslednjo pojavitev, ob kliku na tipko "Zamenjaj" pa bo trenutno izbran niz zamenjal z nizom v "Zamenjaj z" ter poiskal naslednji niz v polju

"Išči". Klik na "Najdi vse" povzroči, da bodo vsi deli besedila, ki ustrezajo iskalnim pogojem, izbrani. To nam omogoča, da na primer poiščemo vse besede 'iskanje' ter jih nato z uporabo trdega oblikovanja (glej razdelek 5.2.6) izpišemo krepko (glej razdelek 5.2.5). S tipko "Zamenjaj vse" z eno potezo zamenjamo vse pojavitve iskalnega niza z novim nizom.

Iskanje se vedno začne pri trenutnem položaju kazalke. Iskanje lahko bolje definiramo z uporabo polj "Možnosti" (slika 5.10):

Samo cele besede izberemo, če iskani vzorec predstavlja celo besedo. OpenOffice.org Writer bo našel besedo `iskanje`, ne bo pa našel besede `vriskanje`.

Nazaj obrne smer iskanja – od trenutne pozicije po dokumentu nazaj.

Regularni izrazi nam omogoča iskanje z uporabo regularnih izrazov, ki namesto konkretnega niza znakov predstavljajo vzorce nizov. Regularni izrazi se zgledujejo po običajni notaciji: "." označuje katerikoli znak; ".*" označuje katerokoli zaporedje znakov; "$" označuje konec odstavka; "^" označuje začetek odstavka, "^$" pa prazen odstavek. Iskanje po iskalnem nizu `.*iskanje` bi kot rezultat vrnilo `vriskanje`, `piskanje` ali `iskanje`.

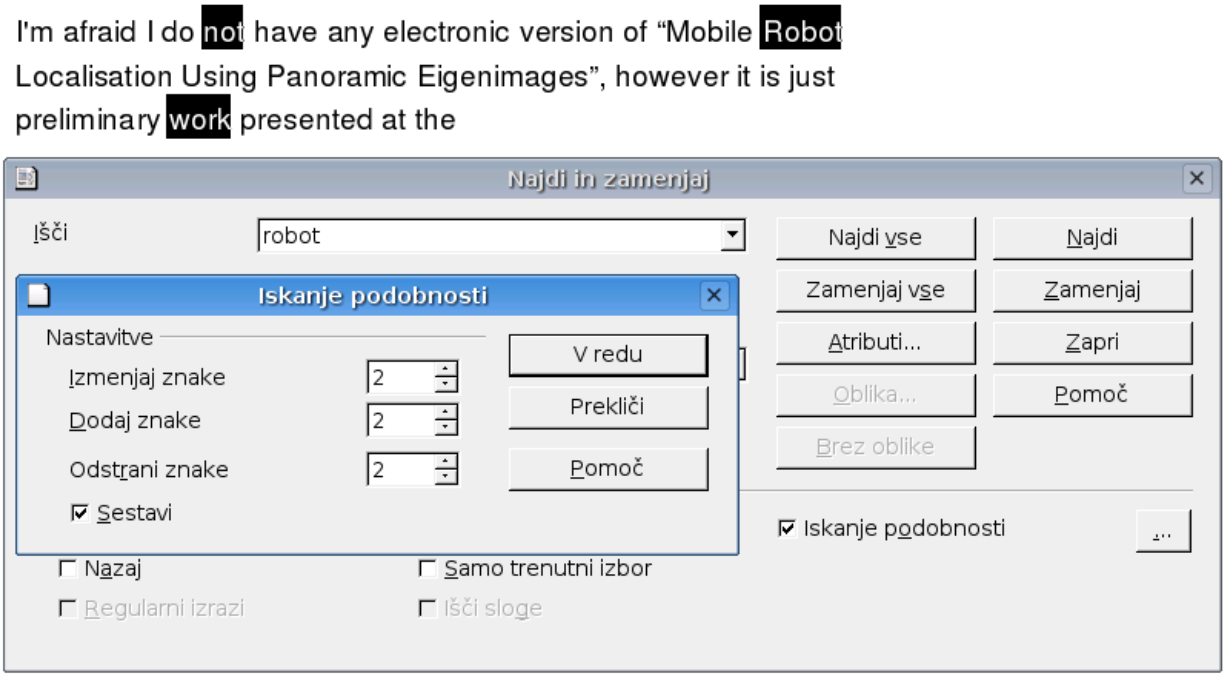
Razlikuj velike in male črke izberemo, ko želimo iskanje, občutljivo na velikost črk. Iskanemu vzorcu `iskanje` bi ustrezalo npr. `vriskanje`, niz `VRISKANJE` pa ne.

Samo trenutni izbor omeji iskanje na trenutno izbrano besedilo.

Išči sloge omogoča iskanje delov besedila s podanim slogom. Možnost je uporabna predvsem za zamenjavo določenih slogov v celem dokumentu. Denimo, da želimo vse dele besedila, izpisane v slogu "Naslov 2", preoblikovati in jim dodeliti slog "Naslov 3". Označimo "Išči sloge", v "Išči" izberemo "Naslov 2", v "Zamenjaj z" pa "Naslov 3" ter kliknemo na gumb *Zamenjaj vse*. Edina omejitev je ta, da lahko iščemo samo sloge odstavkov, ne pa naštevanj ipd.

Iskanje podobnosti omogoča nastavitve parametrov tako, da iskanja ne omejimo le na točne zadetke, temveč dovolimo zamenjave znakov, dodajanje novih in odvzemanje. Primer takega iskanja lahko vidimo na sliki 5.11.

Iskanje lahko dodatno omejimo z iskanjem po atributih, ki niso privzeti (trdo oblikovanje, razdelek 5.2.6). S klikom na tipko "Atributi ..." prikličemo pogovorno okno s spiskom, na katerem izberemo želene atribute. Če nas zanimajo samo nizi, ki so oblikovani specifično, iskalne



Slika 5.11: Iskanje nizov po podobnosti

pogoje nastavimo s klikom na tipko "Oblika ...". Vse dodatne pogoje oblike in atributov razveljavimo s tipko "Brez oblike".

Ko po uspešnem iskanju zapremo pogovorno okno, lahko še naprej krmarimo med pojavitvami iskanega niza (razdelek 5.2.2).

5.2.5 Prvine dokumenta z besedilom

Dokumente z besedilom v grobem sestavljajo naslednje prvine:

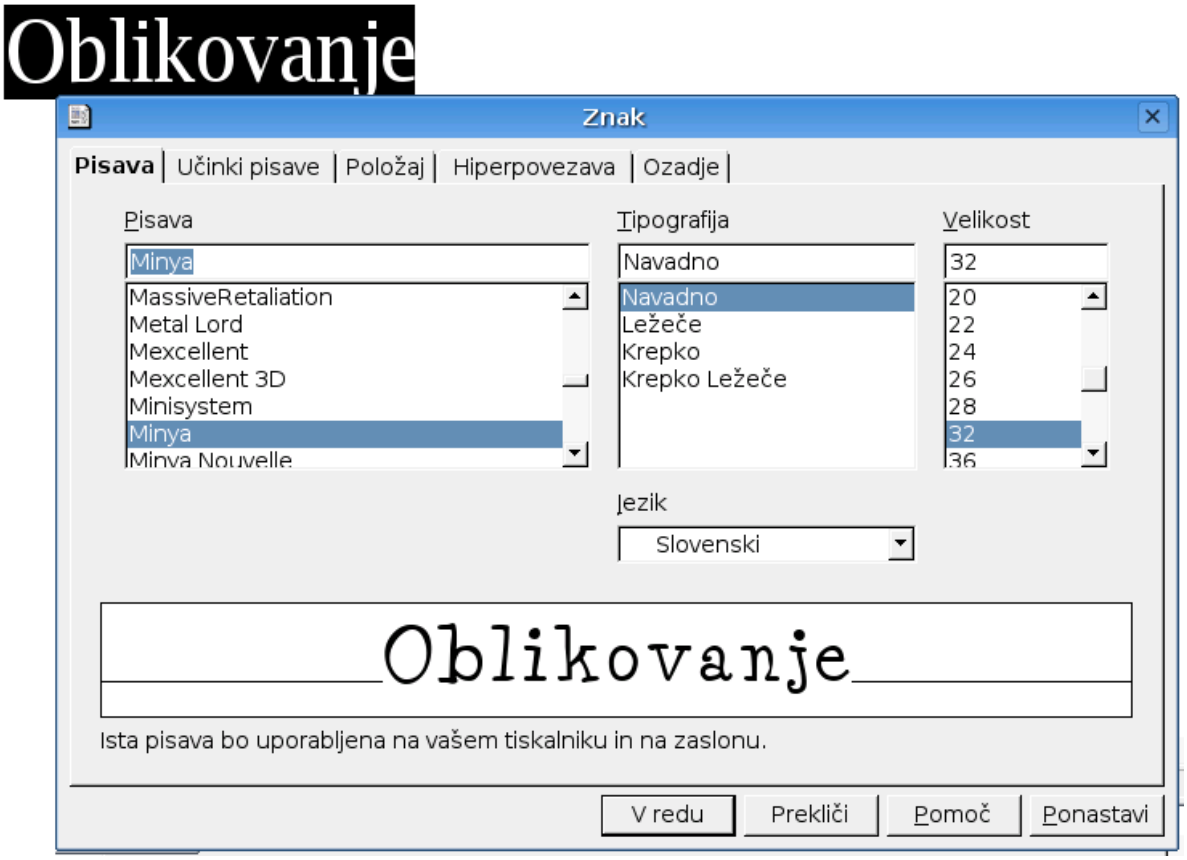
- znaki,
- odstavki,
- odseki,
- stolpci in
- strani.

Poleg naštetih prvin lahko v dokument vstavimo tabele ter grafične ali druge elemente, čemur se bomo posvetili v razdelku 5.2.8.

Vsaka od prvin ima svoj nabor lastnosti, ki prispevajo k videzu dokumenta in jih nastavljamo s pogovornimi okni, dosegljivimi iz menija “Oblika”.

Znaki

Znaki so najosnovnejša prvina pri predstavitvi besedila. Poleg alfanumeričnih znakov (črke in številke) ter ločil lahko v besedilo vnesemo posebne znake, kot so simboli, ter znake, ki jih tiskalnik ne izpiše, vendar imajo svoj pomen pri tem, kako besedilo poteka. Med te tako imenovane nenatisljive znake sodijo presledek, znak za prelom vrstice, znak za konec odstavka ter tabulator.



Slika 5.12: Oblikovanje znakov

Običajno znake vnašamo s tipkovnico. Včasih pa želimo vstaviti simbol, ki ga ne moremo vnesti s tipko ali kombinacijo tipk. Lahko ga vstavimo z uporabo pogovornega okna, ki ga prikličemo z “Vstavi→Posebni znak ...”. Izbiramo lahko med vsemi razpoložljivimi znaki izbrane pisave. Progam običajno ponudi pisavo, s katero trenutno vnašamo besedilo,

vendar lahko v rubriki "Pisava" izberemo poljubno pisavo, ki je nameščena v sistemu. S klikanjem po poljih mreže v osrednjem delu pogovornega okna dodajamo znake v niz, ki ga vidimo v spodnjem levem delu okna. S klikom na tipko "Izbriši" niz izničimo, ko pa smo zadovoljni z njim, kliknemo "V redu" in progam vstavi niz v besedilo.

Če želimo večji nadzor nad znaki, ki jih končni izdelek nima v natisnjeni obliki, vključimo prikaz nenatisljivih znakov s klikom na tipko ¶ v glavni orodni vrstici ali s "Pogled→Nenatisljivi znaki". Program prikaže nenatisljive znake z naslednjimi znaki:

- ¶ prelom odstavka, vstavimo ga s pritiskom na tipko `<Enter>`,
- · presledek,
- ↵ prelom vrstice, ki ne začne novega odstavka, vstavimo ga s kombinacijo tipk `<Shift>`+`<Enter>` ali z "Vstavi→Ročni prelom ...", "Prelom vrstice",
- → tabulator,
- - mesto za delitev besede, ki ga lahko program uporabi, če beseda na koncu vrstice preseže desni rob dokumenta, brez nje pa ostane v vrstici preveč prostora.

Prikaz teh znakov na zaslonu ne bo spremenil videza natisnjenega dokumenta.

Presledek je znak, ki označuje prostor med dvema besedama ali strnjenima nizoma znakov. OpenOffice.org Writer besed in strnjenih nizov ne razdeli med dve vrstici (če ne vklopimo deljenja besed z "Orodja→Deljenje besed ..."), presledke pa vedno uporabi kot mesto deljenja. Če želimo, da določene besede ali strnjeni nizi znakov ostanejo skupaj v vrstici, vstavimo namesto običajnega presledka znak za nedeljivi presledek. To naredimo tako, da držimo tipko `<Control>` in pritisnemo preslednico. Nedeljivi presledki so v dokumentu označeni s sivim ozadjem, podobno kot polja.

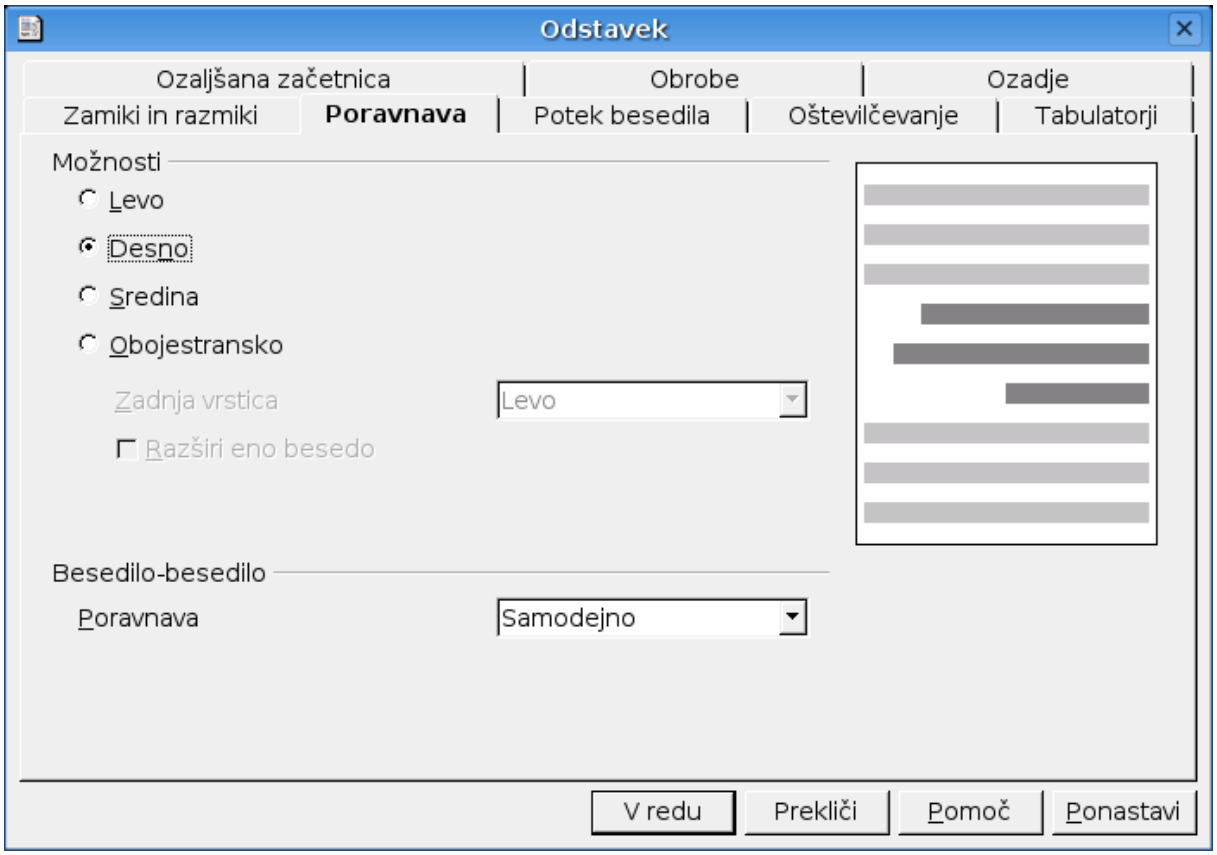
Lastnosti znakov nastavljamo v pogovornem oknu "Oblika→Znak ...". Pojavi se okno s slike 5.12. V njem lahko izbiramo med različnimi pisavami, ki so nameščeni v sistemu, različicami (pokončna, *kurzivna*, **krepka** in ***krepka kurzivna***) ter velikostjo pisave. Učinki vključujejo različne vrste podčrtovanja ali prečrtanega besedila, barve pisave in podobno. Učinek izbranih nastavitev lahko sproti vidimo v spodnjem delu okna. Znakom lahko nastavimo še njihov položaj glede na vrstico ali druge znake, določimo lastnosti hiperpovezave, s čimer postanejo znaki občutljivi na klik, ter ozadje.

Za lažje nastavljanje lastnosti znakov med pisanjem lahko nekatere pogostejše nastavitve nastavljamo v predmetni vrstici. Med te lastnosti spada pisava, velikost pisave, za spreminjanje različice pisave uporabljamo tiple K *I* P (za poljubno kombinacijo **k**repke, *k*urzivne in p̲odčrtane pisave), in barve pisave. Ustrezne rubrike in gumbi te vrstice sproti odražajo lastnosti znakov na mestu kazalke.

Vpliv sprememb lastnosti je odvisna od položaja kazalke in stanja izbora besedila. Če imamo del besedila izbran, se bo sprememba odrazila na označenih znakih. Lastnosti znakov posamezne besede določimo tako, da kazalko postavimo med znake v tej besedi ter nato spremenimo lastnosti znakov. Če pa je kazalka postavljena pred ali za presledek, pa bodo nove lastnosti dobili le tisti znaki, ki jih vnesemo na tistem mestu.

Odstavek

Odstavek je niz znakov, ki jih zaključi znak za nov odstavek (vnesemo ga s tipko `<Enter>`). Večina odstavkov obsega več vrstic, od lastnosti odstavkov pa je odvisno, kako se te vrstice razlivajo med stranmi.



Slika 5.13: Nastavljanje lastnosti odstavka

Lastnosti trenutnega ali izbranih odstavkov nastavljamo v pogovornem oknu, ki ga prikličemo z "Oblika→Odstavek ...". Na strani "Zamiki in razmiki" odstavkom nastavljamo razmik med odstavki, razmik med vrsticami znotraj odstavka ter odmik pred in za besedilom (levi in desni odmik). Posebej lahko nastavimo levi odmik za prvo vrstico, s čimer ustvarimo t.i. viseči zamik odstavka. Na strani "Poravnava" (slika 5.13) določimo levo, desno ali sredinsko poravnavo ter obojestransko, pri kateri program besede v vrstici razporedi tako, da so enakomerno razpostavljene s presledki spremenljive dolžine, prva in zadnja beseda v vrstici pa se dotikata levega oz. desnega roba besedila. Pri tej poravnavi lahko posebej nastavimo še poravnavo zadnje vrstice: leva, desna ali obojestranska. Poravnavo lahko hitro izbiramo tudi izven pogovornega okna s klikom na eno izmed tipk predmetne vrstice. V pogovornem oknu sicer lahko nastavljamo dodatne možnosti za oštevilčevanje odstavkov, določamo tabulatorje, barvo ali vzorec ozadja, obrobe ter ozaljšano začetnico.

Nastavitve zamika besedila ponazarja ravnilo na vrhu okna z dokumentom. Grafična ponazoritev ravnila si svoj videz izposoja pri pisalnih strojih. ÂŠtevilke na ravnilu ponazarjajo odmik od levega roba besedila v izbranih enotah. Spodnji zaznamek označuje levi oz. desni odmik vrstice, zgornji zaznamek pa levi odmik prve vrstice odstavka. Zaznamke lahko vlečemo z miško in s tem nastavljamo ustrezne odmike izbranih odstavkov.

Tabulatorska mesta so posebna mesta na vodoravnem ravnilu, ki določajo, kje se kazalka ustavi pri vstavljanju tabulatorja. S tabulatorji lahko strukturirano oblikujemo odstavke in dosežemo poravnavo delov vrstice, ki niso vezani na poravnavo odstavka.

Slika 5.14: Ravnilo s tabulatorji

Oglejmo si delovanje tabulatorjev na primeru. Če na začetku odstavka vstavimo tabulator s pritiskom na tipko `<Tab>`, se bo kazalka premaknila v desno za daljšo razdaljo. Ustavila se bo na prvem mestu, ki je na ravnilu označeno z . Tabulator je torej presledek v dolžini od mesta, kjer smo znak vnesli, do prvega naslednjega tabulatorskega mesta. Poskusimo na konec trenutne vrstice vstaviti besedo `desno`. Nato pritisnimo tipko `<Home>`, ki nas prestavi na začetek vrstice, in vtipkajmo besedo `levo`. Vidimo, da beseda `desno` ostane na mestu, čeprav pred njo vrivamo novo besedilo. Če nato pred tabulatorjem natipkamo še nekaj besed tako, da se z leve približamo besedi `desno`, se bo ta beseda pomaknila do naslednjega tabulatorskega mesta.

Privzeta tabulatorska mesta so enakomerno razvrščena po vrstici, razmik med njimi pa je določen globalno za vse dokumente, ki jih odpremo v svojem OpenOffice.org Writerju. Razmik med tabulatorskimi mesti lahko nastavimo v "Orodja→Možnosti ...", na levi izberemo "Dokument z besedilom→Splošno" in spremenimo vrednost v polju "Tabulatorska mesta". Videz dokumenta, ki vsebuje tabulatorje, se torej lahko razlikuje na zaslonu in tiskalniku drugega računalnika. Zato pred uporabo tabulatorjev raje nastavimo fiksna tabulatorska mesta.

Tabulatorska mesta nastavljamo na strani "Tabulatorji" v pogovornem oknu, ki ga prikličemo z "Oblika→Odstavki ...". Izbiramo lahko med štirimi vrstami tabulatorjev, ki se med seboj razlikujejo po poravnavi. Besedilo je torej poravnano glede na vrsto tabulatorskega mesta, kjer se tabulator ustavi, in sicer:

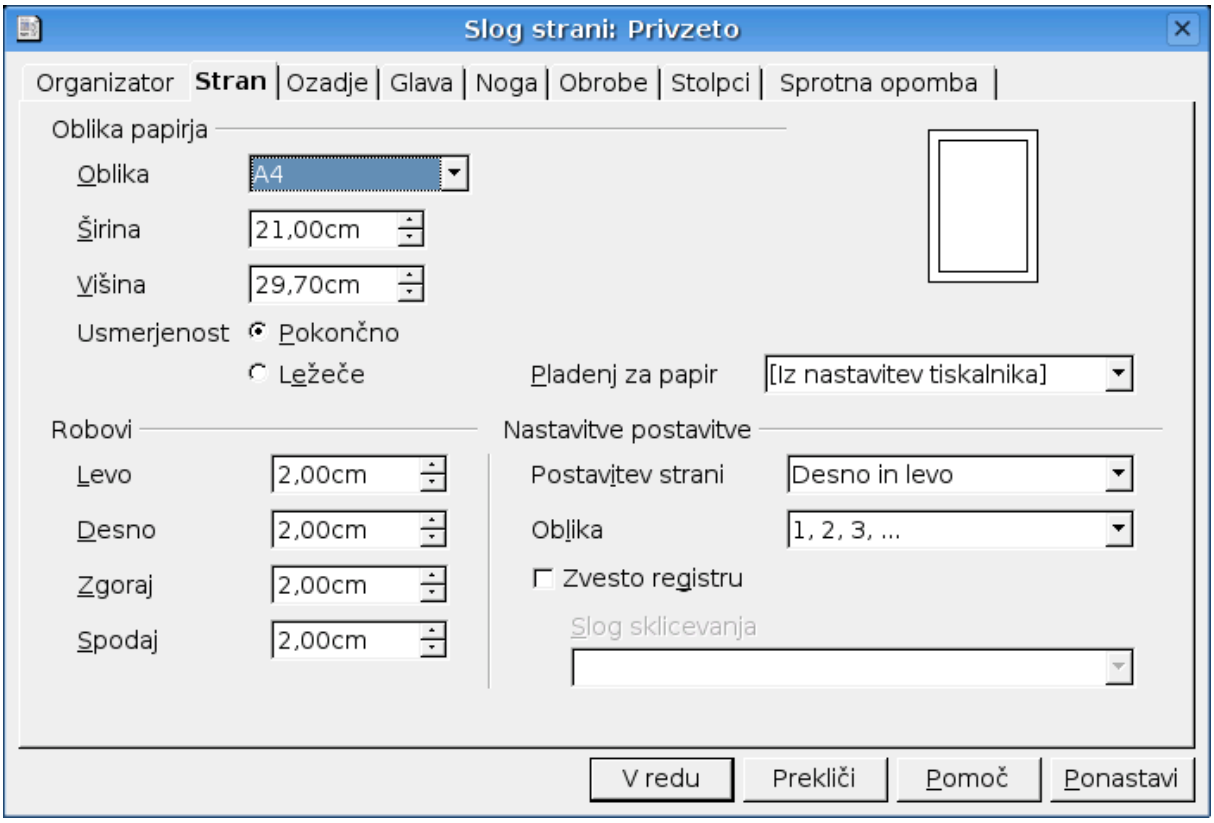
- leva poravnava: besedilo se izpisuje desno od tabulatorskega mesta,
- desna poravnava: besedilo se izpisuje levo od tabulatorskega mesta,
- decimalna poravnava: besedilo (številke) pred decimalno vejico se izpisuje v levo, za decimalno vejico pa v desno; namesto decimalne vejice lahko v polje "znak" vnesemo poljuben znak, ki bo v besedilu prelomnica med desno in levo poravnavo,
- sredinska poravnava.

S tabulatorskimi mesti preprosto upravljamo na vodoravnem ravnilu (slika 5.14), ki prikazuje mesto in vrsto tabulatorskega mesta. S klikom na prazni del ravnila dodamo mesto, katere vrsto vidimo v okvirčku na levi strani ravnila. Vrsto mesta izbiramo s klikanjem znotraj okvirčka. Obstoječe oznake vlečemo z miško po ravnilu. Če želimo mesto odstraniti, oznako potegnemo stran od ravnila.

Strani, stolpci in odseki

Ciljni medij dokumenta z besedilom je običajno papir. Ko pripravljamo dokument, prilagodimo lastnosti strani dimenzijam papirja. Glede na estetske zahteve ter tehnične lastnosti strani določimo robove besedila, znotraj katerih poteka besedilo. Stranem lahko nastavljamo tudi usmerjenost, lihe in sode strani pa se lahko razlikujejo v nekaterih lastnostih.

Levi in desni rob besedila lahko nastavljamo tako, da z miško vlečemo oznake robov na ravnilu (paziti moramo, da miško namestimo na sredino med zaznamka za levi odmik prve in ostalih vrstic; na pravem mestu smo



Slika 5.15: Lastnosti strani

takrat, ko kazalec miške dobi obliko ↔). Večji nadzor nad številskimi vrednostmi dobimo v pogovornem oknu, ki ga prikličemo z "Oblika→Stran ...".

OpenOffice.org Writer sam doda novo stran, ko besedilo napolni trenutno stran. Dokument bo torej dolg ravno toliko strani, kolikor jih besedilo potrebuje. Če želimo prehod na novo stran, preden napolnimo trenutno stran, vstavimo ročni prelom strani s pritiskom na kombinacijo tipk `<Control>`+`<Enter>` ali če v pogovornem oknu "Vstavi→Ročni prelom ..." izberemo "Prelom strani". Če želimo, da se s prelomom spremenijo lastnosti strani, iz padajočega menija "Slog" izberemo ustrezen slog strani (o slogih več v razdelku 5.2.6).

Besedilo lahko poteka v več stolpcih. ÂŠtevilo stolpcev, njihovo postavitev in medsebojne razmike nastavljamo v pogovornem oknu "Oblika→Stolpci ...". Ko potrdimo spremembe, bo celoten dokument razdeljen po stolpcih. Program samodejno razporedi besedilo po stolpcih od leve proti desni na vsaki strani. Nastavitve stolpcev odraža tudi ravnilo, s katerim lahko grafično urejamo širino posameznih stolpcev ter razmik med stolpci. Zaznamke za odmike vrstic vidimo le za izbrane odstavke.

Vsebino posameznega stolpca lahko urejamo samo, če besedilo doseže ta stolpec. Predčasni prehod na nov stolpec vstavimo tako, da izberemo "Vstavi→Ročni prelom ..." in izberemo "Prelom stolpca".

Enostavnim dokumentom lahko v celoti nastavimo lastnosti strani ali stolpcev. V kompleksnejših dokumentih pa želimo za posamezne dele te lastnosti nastavljati ločeno. Vsak tak del imenujemo odsek. Na mesto, kjer želimo preiti na nov odsek, postavimo kazalko in izberemo "Vstavi→Odsek ...". V rubriko "Nov odsek" vnesemo ime odseka ter nastavimo ustrezne lastnosti.

5.2.6 Oblikovanje dokumenta

Ko pripravljamo dokument z besedilom, moramo biti pozorni na vsebino, ta pa zares zaživi šele z obliko. S pravilnim oblikovanjem lahko dosežemo estetski videz, ki pa se mora podrejati funkcionalnim lastnostim posameznih delov besedila.

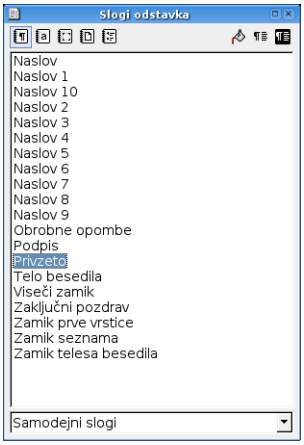
Poznamo dve vrsti oblikovanja. **Trdo oblikovanje** vpliva neposredno na izbrani del besedila. **Mehko oblikovanje** pa deluje na enotne dele besedila v celotnem dokumentu, ki jim dodelimo funkcionalni pomen. Za takšno oblikovanje uporabljamo **sloge**.

Trdo oblikovanje

Trdo oblikovanje je vsako določanje lastnosti izbranega besedila – znakov, odstavkov in drugih elementov. Uporabimo ga za krajše dele besedila, za katere želimo, da se razlikujejo od preostalega besedila. Najpogosteje s trdim oblikovanjem poudarimo določene besede ali zaporedja besed (s krepko ali kurzivno pisavo, z drugo barvo ...) oziroma če želimo hitro pripraviti krajši dokument. Pri delu z daljšimi dokumenti pa je bolj smotrno uporabjati sloge, čeprav je na videz enostavneje besedilo oblikovati trdo.

Mehko oblikovanje s slogi

Vsako resno besedilo je strukturirano na več funkcionalnih enot, na primer naslov, kazalo, povzetek, naslovi poglavij, podnaslovi ter telo besedila. Vsaka funkcionalna enota se lahko loči od ostalih enot v vizualnih lastnostih, pri čemer je enotno oblikovanje v besedilu izredno pomembno. **Slogi** nam omogočajo, da med vnosom in urejanjem delom besedila dodelimo pomen, pripadajoče besedilo pa bo dobilo obliko, ki je določena v slogovniku. Če se odločimo za spremembo oblike funkcionalne enote besedila, spremenimo lastnosti pripadajočega sloga, spremembe pa se bodo takoj odrazile povsod v dokumentu, kjer smo slog uporabili.



Slika 5.16: Slogovnik

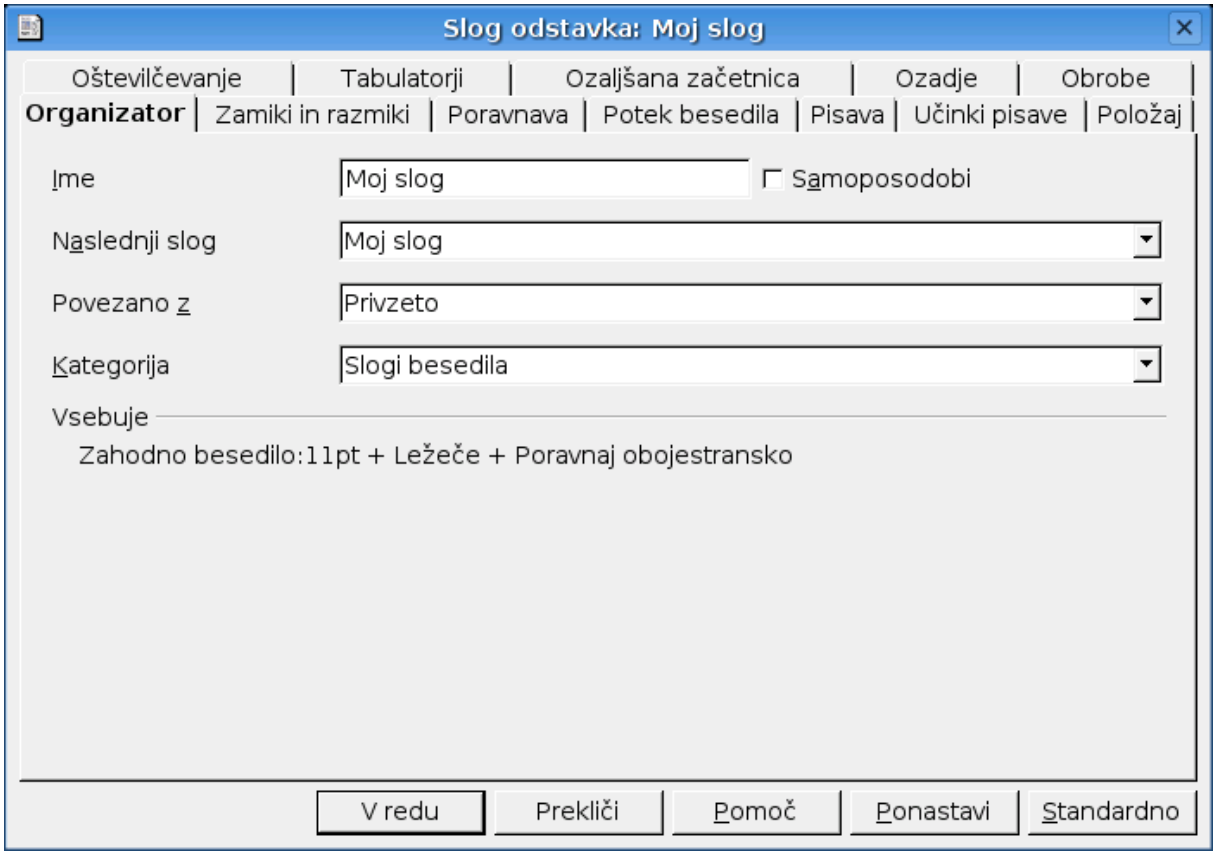
Ime sloga, ki je dodeljen besedilu na mestu kazalke, lahko vidimo v predmetni vrstici levo od imena izbrane pisave. S tem padajočim menijem lahko zamenjamo slog, vendar meni običajno vsebuje le imena slogov, ki smo jih uporabili v dokumentu. Celoten seznam slogov lahko vidimo v plavajočem oknu, ki ga prikličemo s pritiskom na tipko `<F11>`, z gumbom v funkcijski vrstici oz. vklopimo možnost "Oblika→Slogovnik". Na zaslonu vidimo okno, kot je tisto na sliki 5.16.

OpenOffice.org Writer pozna pet vrst slogov, vsaka vrsta pa je vezana na prvino dokumenta, katere obliko določa. V zgornjem levem delu okna na sliki 5.16 lahko določimo, katero vrsto slogov naj vsebuje seznam v osrednjem delu okna. Izbiramo med slogi odstavka, znakov, slogi okvirov z besedilom, grafiko ipd., strani in oštevilčenja. Ker je seznam lahko dolg, je na dnu okna padajoči meni s kriterijem za prikaz imen slogov v seznamu. Ravnanje s slogi različnih vrst je podobno, zato bomo opisali samo postopek za delo s slogi odstavka. Zato v zgornjem delu kliknimo, spodaj pa izberimo možnost "Slogi besedila".

Sloge odstavku dodeljujemo tako, da odstavek izberemo ali postavimo vanj kazalko. Če želimo dodeliti slog, ki ga v dokumentu še nismo uporabili, na seznamu dvokliknemo ime sloga. Ob nadaljnji uporabi lahko slog izberemo iz padajočega menija v predmetni vrstici. V vsakem primeru besedilo takoj dobi obliko dodeljenega sloga.

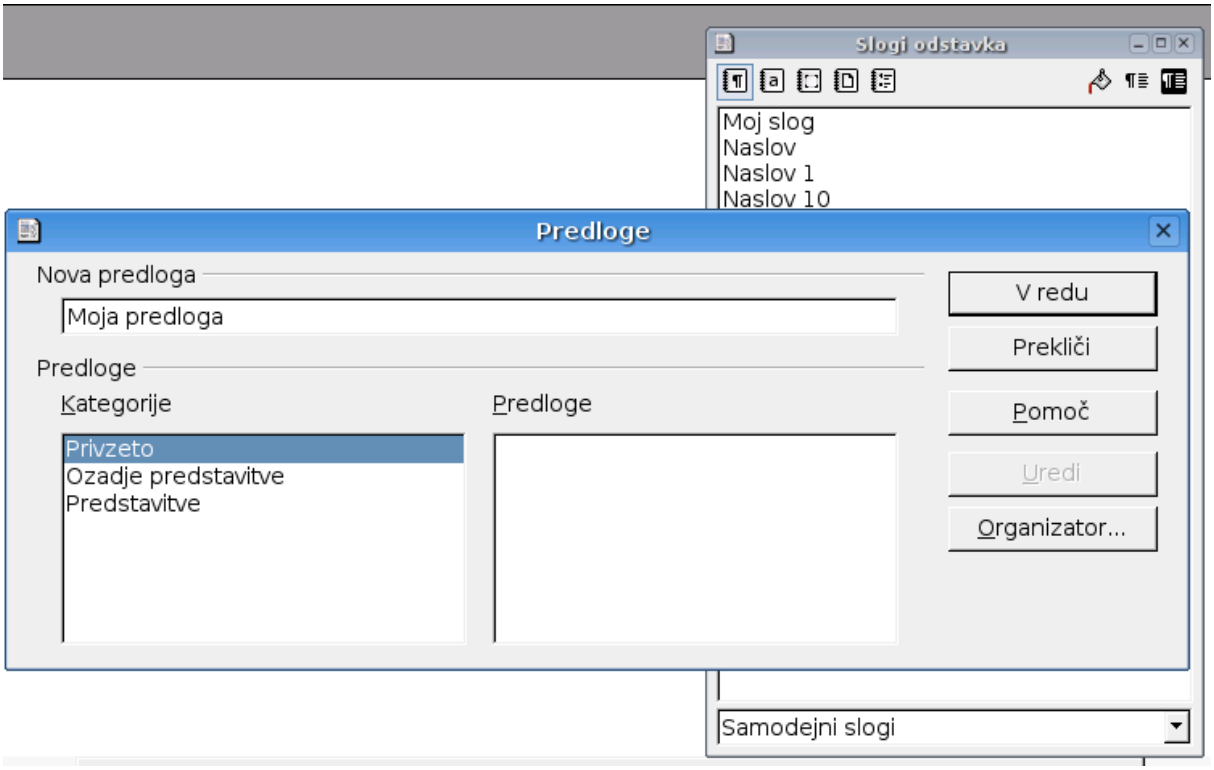
Ko ustvarimo nov dokument, ima običajno odstavek dodeljen slog "Privzeto". Vnesimo naslov poglavja (npr. "Urejanje besedil") in še preden pritisnemo `<Enter>`, dodelimo odstavku slog "Naslov 1". Besedilo odstavka bo dobilo obliko, ki nakazuje prvi naslov v hierarhiji. Ko pritisnemo `<Enter>`, začnemo z novim odstavkom, ki pa bo imelo slog "Telo besedila". Čeprav morda nepričakovana, je ta sprememba smiselna, saj običajno naslovu sledi telo besedila. Če želimo dodati podnaslov,

vnesemo besedilo podnaslova in nastavimo slog “Naslov 2”. Višja številka naslova pomeni večjo stopnjo podrejenosti v hierarhiji naslovov. Seveda pa so naslovi le en primer uporabe slogov, saj drugi deli besedila lahko uporabijo svoje sloge. Vendar imajo slogi naslovov še dodaten pomen, saj omogočajo programu samodejno izgradnjo kazala (razdelek 5.2.10).



Slika 5.17: Ustvarjanje novega sloga z organizatorjem slogov

Lastnosti slogov lahko spreminjamo v katalogu slogov (“Oblika→Slogi→Katalog slogov . . . ”) ali tako, da v slogovniku kliknemo z desno tipko miške na ime sloga ter izberemo “Spremeni. . . ”. Pojavi se pogovorno okno s slike 5.17, v katerem na strani ”Organizator“ nastavimo ime sloga, izberemo naslednji slog (tj. slog, ki ga dobi odstavek, ki ga vstavimo neposredno za odstavek s tem slogom), v spodnjem delu strani pa vidimo rubriko “Vsebuje”, ki prikazuje seznam posebej za ta slog nastavljenih lastnosti. V rubriki “Povezano z” izberemo slog, po katerem trenutni slog prevzame lastnosti, ki jih sam ne določa neposredno. Sprva je vrednost tega polja enaka slogu, ki smo ga označili v slogovniku, preden smo ustvarili nov slog. Na ostalih straneh pogovornega okna nastavljamo lastnosti znakov in odstavka na podoben način, kot smo to že spoznali. Razlika je le ta, da lahko določene lastnosti pustimo nenastavljene, s



Slika 5.18: Shranjevanje nove predloge

čemer dopustimo, da besedilo ohrani lastnosti, nastavljene v nadrejenem slogu.

Ko potrdimo spremembe lastnosti sloga, lahko v dokumentu te spremembe takoj vidimo povsod, kjer smo uporabili ta slog. Slogi so torej izredno močno orodje, pri delu z daljšimi in kompleksnejšimi dokumenti pa je njihova uporaba nujna, saj bomo v nasprotnem primeru zelo verjetno porabili preveč časa za operacije, ki jih progam zmore opraviti samodejno.

Spremembe v slogih ter nanovo ustvarjene sloge program samodejno shrani skupaj z dokumentom, v katerem smo sloge spreminjali. Če želimo sloge prenašati med dokumenti, v katalogu slogov kliknemo na tipko "Organizator ..." in v pogovornem oknu, ki se nam odpre, izvedemo prenose. Če pogosto ustvarjamo dokumente, ki imajo določeno obliko, je smiselno sloge prenesti v prazen dokument, ki ga nato shranimo kot predlogo dokumenta z besedilom (razdelek 5.2.7).

5.2.7 Predloge dokumentov in čarovniki

Omenili smo že, da lahko dokument shranimo kot predlogo, ki jo kasneje uporabimo za ustvarjanje novega dokumenta s slogi in obliko iz predloge.

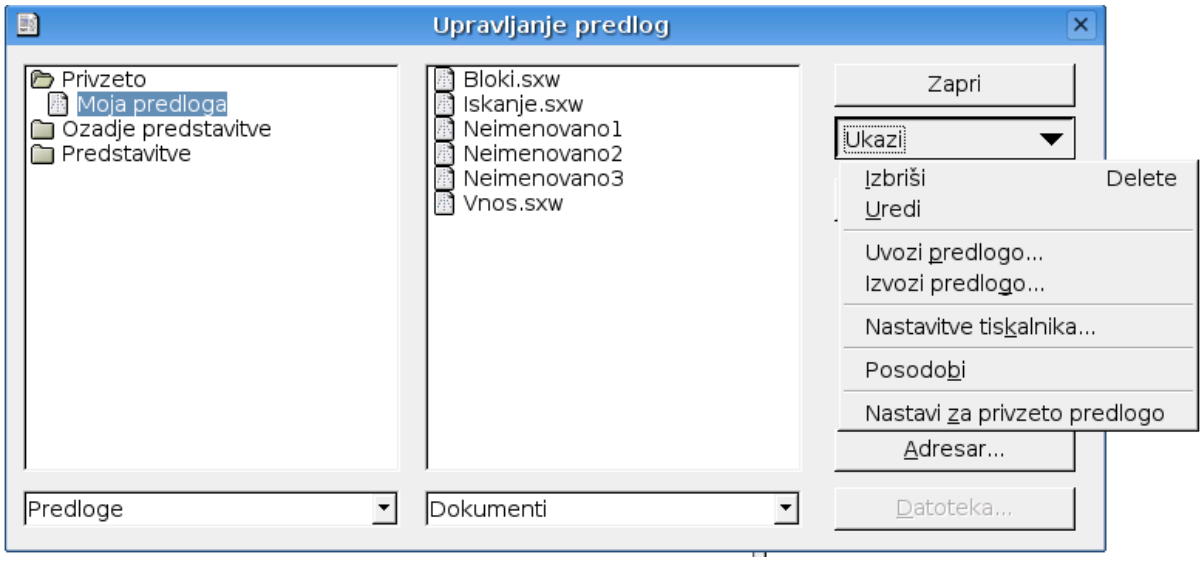
Pripravimo si lahko predloge za vrste dokumentov, ki jih pogosto pišemo, npr. za dopis, poročilo o delu ali vabilo. Predloge so lahko prazne ali vsebujejo enotno besedilo. Na mestih, kjer predvidimo izpis datuma, ure, avtorja ali drugih podatkov, ki se med dokumenti spreminjajo, lahko vstavimo ustrezno polje, ki se ob ustvarjanju dokumenta osvežijo ("Vstavi→Polja→...").

Predlogo lahko ustvarimo iz vsakega dokumenta, ki smo mu dodali ali spremenili lastnosti sloga. Ohranimo lahko generično vsebino, ki se bo pojavila v vsakem novo ustvarjenem dokumentu, ali pa pustimo dokument prazen in s tem ohranimo le sloge in oblikovanje. Predlogo nato shranimo tako, da bodisi v "Datoteka→Predloge→Shrani..." (slika 5.18) vnesemo ime nove predloge, bodisi dokument posnamemo z "Datoteka→Shrani kot..." in izberemo vrsto dokumenta "OpenOffice.org 1.0 Predloga dokumenta z besedilom" ("OpenOffice.org 1.0 Text Document Template"). V prvem primeru bo program datoteko z vzorcem shranil v uporabniško področje map programov OpenOffice.org in dodal ime predloge na spisek vidnih predlog. Tako bomo shranili sloge, ki jih želimo uporabljati sami na trenutnem računalniku. V drugem primeru dobimo datoteko z vzorcem v izbrani mapi, na spisek predlog pa program predloge ne doda.

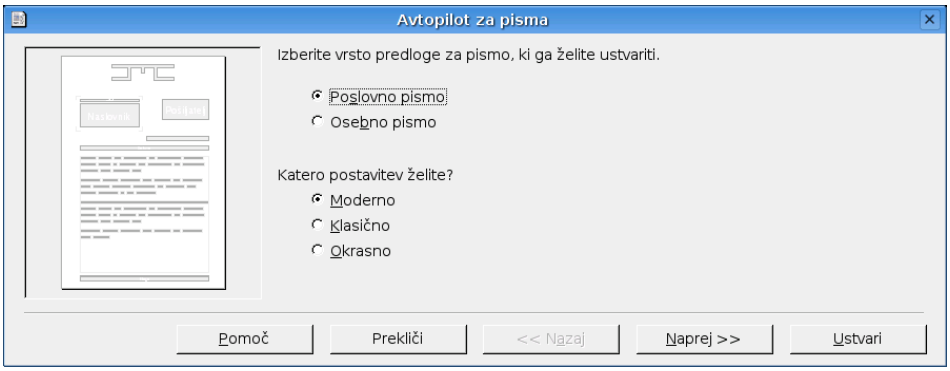
Dokument, ki ga odpremo s klikom na ikono *Nov dokument z besedilom* ali pa z izbiro "Datoteka→Nov dokument z besedilom", uporablja privzeto predlogo. Če nam privzete nastavitve ne ugajajo, jih lahko spremenimo tako, da posežemo v njegove nastavitve. Privzete nastavitve za pisavo lahko spremenimo v "Orodja→Možnosti→Dokument z besedilom→Osnovne pisave". V pogovornem oknu, ki ga prikličemo z "Datoteka→Predloge→Organiziraj..." lahko iz spiska "Predloge" izberemo predlogo, nato iz menija "Ukazi" izberemo "Nastavi za privzeto predlogo" (slika 5.19).

Nov dokument ustvarimo iz vzorca tudi tako, da izberemo "Datoteka→Nova→Predloge in dokumenti".

Pri ustvarjanju novih dokumentov in vzorcev si lahko pomagamo tudi s čarovnikom, imenovanim Avtopilot, ki nas vodi skozi sistem izbirnih zaslonov. Prikličemo ga z "Datoteka→Avtopilot→(vrsta dokumenta)". Vrsta dokumenta je lahko pismo, faks, dnevni red, okrožnica itd. V izbirnih zaslonih, nato definiramo polja v dokumentu, njihovo razporeditev in splošen videz dokumenta. Primer avtopilota vidimo na sliki 5.20. Končni rezultat oblikovanja s čarovnikom lahko shranimo tudi kot nov vzorec.



Slika 5.19: Upravljanje predlog



Slika 5.20: Ustvarjanje dokumenta s čarovnikom

5.2.8 Vstavljanje vizualnih elementov

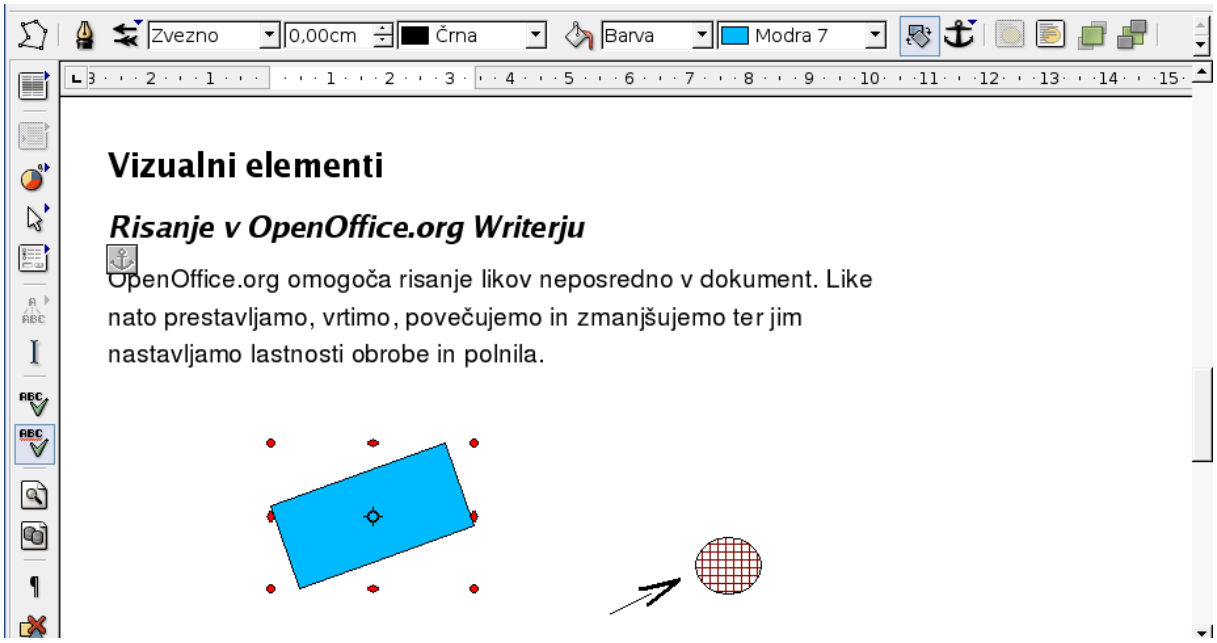
Liki in grafika

Dokument z besedilom postane nazornejši in privlačnejši, če ga opremimo s slikovnim gradivom, grafi, tabelami in drugimi elementi. OpenOffice.org Writer omogoča enostavno dodajanje in vizualno urejanje tako enostavnih likov kot kompleksnih predmetov, ki jih pripravimo z drugim programom.

Slika 5.21: Orodja za risanje

Enostavne like lahko v dokument dodamo z orodji, vgrajenimi v OpenOffice.org Writer. Do njih pridemo tako, da na glavni orodni vrstici kliknemo in držimo gumb "Pokaži funkcije risanja" . Odprla se bo plavajoča orodna vrstica s funkcijami risanja, kot je tista na sliki 5.21. Po kliku na enega izmed gumbov vrstica izgine, če pa kliknemo na njegovo naslovno vrstico, ostane viden na mestu, kamor ga odvlečemo. Med drugim ponuja orodje "Izbor", s katerim označujemo in izbiramo obstoječe slikovne elemente, ravne črte, pravokotnike, krožnice in kroge, krivulje ter mnogokotnike. Ko izberemo orodje, nam videz kazalca miške pokaže, da lahko začnemo z risanjem, na mestu gumba v glavni orodni vrstici pa dobimo gumb zadnjega uporabljenega orodja za hitrejši dostop. Like rišemo enako kot v risarskem programu OpenOffice.org Draw.

Obstoječi lik v dokumentu izberemo tako, da nanj kliknemo. Okrog lika se pojavijo točke, ki jih lahko vlečemo z miško in s tem spreminjamo lik. Vrsto spremembe nakaže oblika miškinega kazalca, ko ga zapeljemo nad posamezno točko. Lik lahko premikamo po dokumentu tako, da z miško zagrabimo njegovo notranjost in ga vlečemo po besedilu. Mesto, kamor je izbrani lik vstavljen, prikazuje ikona sidra. Primer videza dokumenta na zaslonu med urejanjem likov vidimo na sliki 5.22.



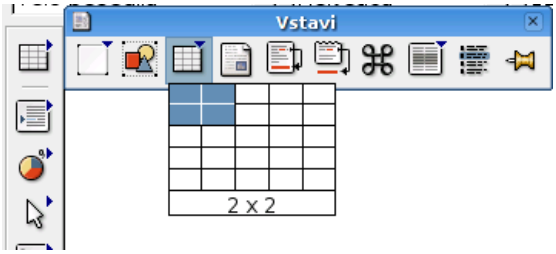
Slika 5.22: Slike in risbe v dokumentu z besedilom

Bitne slike vstavimo z "Vstavi→Grafika→Iz datoteke..." in poiščemo datoteko, ki hrani sliko. slika se nato pojavi v dokumentu na privzetem mestu, njeno velikost in lokacijo pa spreminjamo podobno kot to delamo z liki.

Predmetna vrstica odraža vrsto in lastnosti predmeta, ki ga izberemo. Za like so te lastnosti oblika, barva, debelina obrobe, barva in vzorec polnila itd. Pri bitnih slikah lahko nastavljamo lastnosti, kot so korekcija barvnih kanalov, svetlost, kontrast, faktor gama, prosojnost, določamo lahko tudi nekaj učinkov.

Glede na izbrani predmet se spreminjajo možnosti v meniju "Oblika". Večina možnosti oblike je dostopna tudi v priročnem meniju, ki ga prikličemo z desnim klikom na lik ali predmet. V podmeniju "Razporedi" (pri bitnih slikah je to podmeni "Prilagodi") spreminjamo vrstni red prikaza likov, kar je uporabno, če predmeti delno ali v celoti prekrivajo druge predmete; "Poravnava" je aktivna le, če izberemo več predmetov, omogoča pa poravnavo likov glede na njihove medsebojne odnose; v podmeniju "Oblivanje" določimo, kako naj se besedilo v bližini predmeta preliva okrog predmeta; v podmeniju "Zasidraj" izbiramo, ali naj se predmet zasidra na stran, vrstico ali kot znak.

Tabele

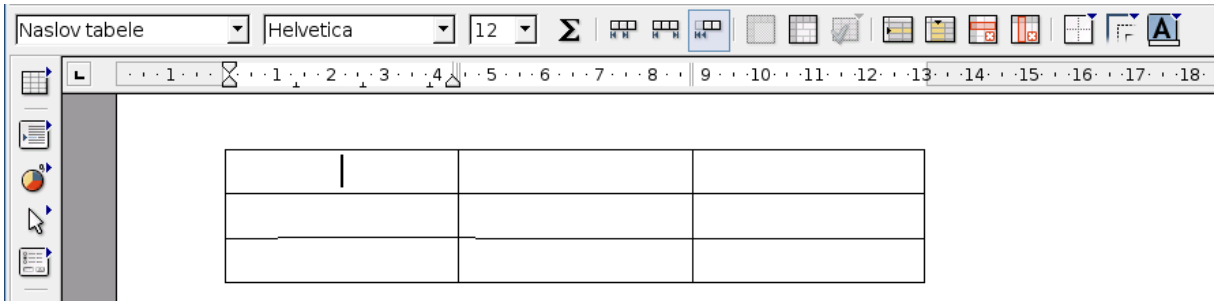


Slika 5.23: Določanje dimenzij tabele v orodni vrstici pred vstavljanjem

S tabelo na strukturiran način prikažemo podatke ali odnose med podatki. Tabelo vstavimo z "Vstavi→Tabela..." ali s klikom na gumb "Vstavi tabelo" iz orodjarne "Vstavi" glavne orodne vrstice. V pogovornem oknu, ki se nam odpre, nastavimo število vrstic in stolpcev ter druge lastnosti tabele, ki jo vstavljamo v dokument. Če želimo tabelo vstaviti na bolj vizualen način, v glavni orodni vrstici odpremo orodjarno "Vstavi" in zadržimo miško na gumbu "Vstavi tabelo". Pod gumbom se bo pojavila shema tabele, na katero lahko premaknemo miško in s tem določimo, kako razsežno tabelo želimo vstaviti (slika 5.23).

V celice tabele vnašamo vrednosti kot običajno besedilo, med sosednimi celicami pa se premikamo s tipko `<Tab>` oz. `<Shift>`+`<Tab>`.

Ko se kazalka nahaja v celici tabele ali imamo izbran del tabele, odraža predmetna vrstica nekatere lastnosti celice oz. tabele ter vsebuje ukaze za združevanje izbranih celic v eno celico, razdeljevanje celice, vstavljanje in brisanje vrstic in stolpcev ter nastavitev barv in drugih lastnosti obrob in



Slika 5.24: Predmetna vrstica za tabelo

ozadja celic (slika 5.24). ÂŠe več možnosti za nastavitev lastnosti celic in tabele najdemo v priročnem meniju ter v pogovornem oknu "Oblika→Tabele..."

Čeprav pri vstavljanju tabele nismo določili nobenih oblikovnih lastnosti, imajo celice že nastavljene sloge odstavka: prva vrstica dobi slog Naslov tabele, ostale celice pa Vsebina tabele.

Namiznozaložniške tehnike

Oblikovalci si večkrat pomagajo s stavljenjem besedila v okvirje. Tako besedilo poudarijo ali ločijo od ostale vsebine. Stran tudi razdelijo na več stolpcev, kar olajša branje (razdelek 5.2.5).

Okvir vnesemo v dokument z "Vstavi→Okvir...". V pogovornem oknu nastavimo obliko okvira, nato pa v sam okvir vnesemo besedilo. Primer dokumenta s tremi stolpci in okvirjem vidimo na sliki 5.25.

Predmeti ostalih programov

OpenOffice.org uporablja protokol OLE (*Object Linking and Embedding*), ki omogoča izmenjavo med dokumenti. V OpenOffice.org Writer lahko vstavimo katerikoli dokument, ustvarjen v drugem programu zbirke OpenOffice.org, npr. preglednico iz OpenOffice.org Calca. Predmet OLE vnesemo z izbiro "Vstavi→Predmet→Predmet OLE...". V pogovornem oknu izberemo tip predmeta. Odpremo lahko predmet iz datoteke ali pa ustvarimo novega. Datoteka novega predmeta je sedaj povezana z našim dokumentom. Pomembno je vedeti, da lahko to dosežemo le z uporabo protokola OLE, ne pa z običajnim vstavljanjem predmetov, saj slednje ne zagotavlja dinamične povezave.


Ko so se konec 70-ih let pojavili tako imenovani "miniračunalniki", so se začeli razvijati tudi programi za računalniško urejanje

Urejanje besedil

besedil. Osebni računalniki pa so v začetku 90-ih let pri pisanju in urejanju besedil praktično povsem izrinili klasične pisalne stroje. Prednosti urejanja besedil z računalnikom so očitne. Z njimi lahko hitreje in laže spreminjamo, popravljamo in preoblikujemo besedila. Prvi računalniški programi za oblikovanje besedil so bili

kombinaciji s slikovnim gradivom (npr. program PageMaker). Ker so danes tako rekoč vsakemu uporabniku dostopna zelo zmogljiva orodja za oblikovanje besedil, je težko potegniti ločnico med običajnim oblikovanjem kratkega besedila, kot je na primer poslovno pismo, in zahtevnim tipografskim oblikovanjem, kot ga zahteva neka knjiga. Začetniki grešijo predvsem pri pretirani uporabi in nesmotrni kombinaciji raznovrstnih možnosti, ki jih ta orodja omogočajo. Na primer, z uporabo

--- What You See Is What You Get) in

- **logično** urejanje besedil

Vizualno urejanje besedil

Osnovni princip vizualnega urejanja besedil je, da so rezultati vseh uporabnikovih akcij takoj vidni na zaslonu. Programi za vizualno urejanje besedil zahtevajo sicer sodobno strojno opremo in grafični uporabniški vmesnik, vendar pa so enostavni za uporabo in se jih je možno hitro naučiti, saj je delo z njimi konceptualno zelo podobno pisanju na pisalnem stroju. Hkrati ko pišemo besedilo,

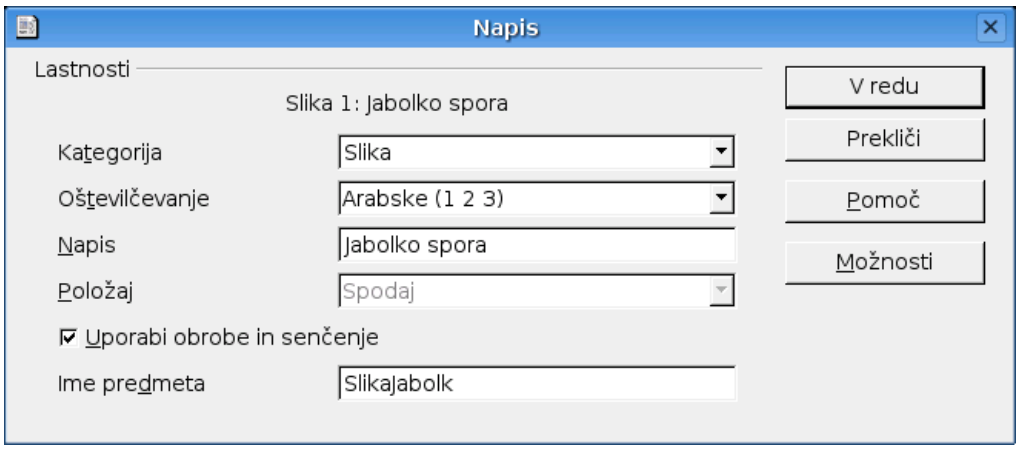
Slika 5.25: Dokument s tremi stolpci in pravokotnikom, ki vsebuje besedilo, meče senco in ima oblivanje nastavljeno za likom ter razmikom 0,20 cm v vsaki smeri

5.2.9 Opremljanje vizualnih elementov in sklicevanje

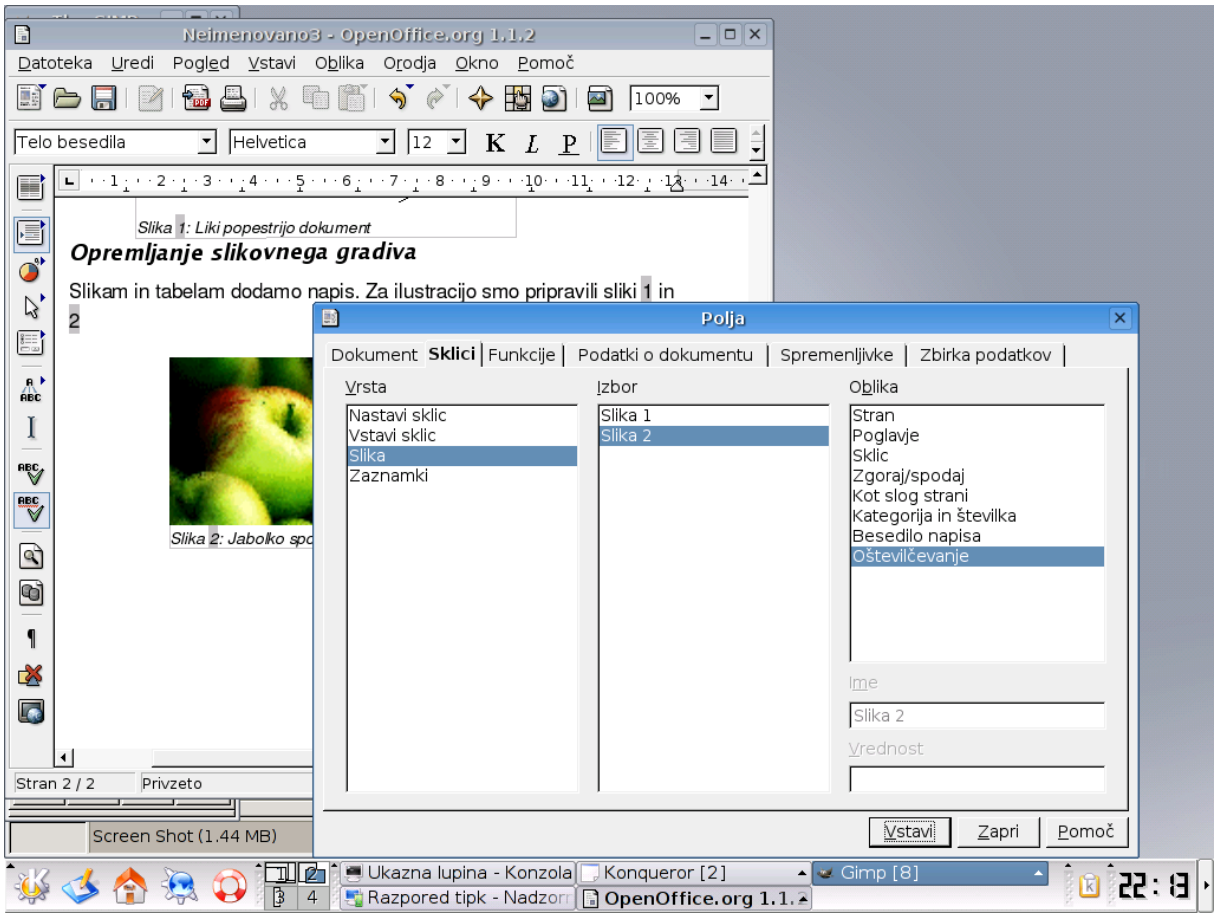
Vizualni elementi v dokumentu potrebujejo razlago ali naslov. Razlago slike, tabele risbe ipd. vnesemo v samo besedilo, sam vizualni element pa opremimo z napisom. Elementi imajo zaporedne številke glede na vrstni red njihove pojavitve v dokumentu. Ročno številčenje je nesmotrno, saj vodi do napak, ko spremenimo dokument, oznak pa ne. Zato je najbolje številčenje prepustiti programu, pri delu s sklici pa uporabljati polja, ki jih bo program za prikaz in tisk nadomestil s pravilnimi vrednostmi. Poleg tega lahko elemente z napisi izpišemo v kazalu slik, tabel ali drugih elementov (razdelek 5.2.10).

V OpenOffice.org Writerju dodamo vizualnemu elementu napis tako, da kliknemo na element z desno tipko in iz priročnega menija izberemo "Napis ...". Izberemo ali vnesemo kategorijo, obliko številčenja, v polje "Napis" pa vnesemo besedilo, ki bo opremilo element v dokumentu (slika 5.26).

V besedilu se pogosto želimo sklicevati ali bralca usmeriti na nek vizualni element, pa tudi na drug element, kot je stran, poglavje ipd. V OpenOffice.org Writerju to naredimo z "Vstavi→Navzkrižno sklicevanje...". Okno, ki se pojavi in je podobno tistemu na slik 5.27, za razliko od večine ostalih pogovornih oken ne blokira dostopa do dokumenta. Zato lahko prestavljamo kazalko po dokumentu. Iz seznama



Slika 5.26: Dodajanje napisa pod sliko



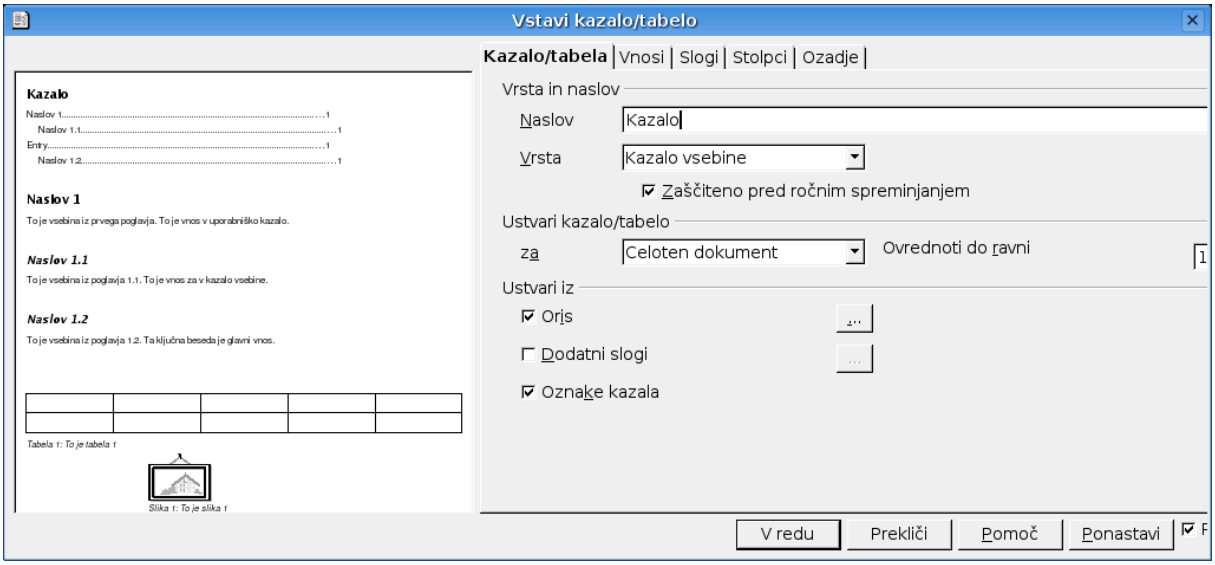
Slika 5.27: Navzkrižno sklicevanje

"Vrsta" izberemo vrsto elementa, na katerega se želimo sklicevati. Seznam "Izbor" nato prikaže spisek elementov izbrane vrste v trenutnem dokumentu. Iz seznama "Oblika" pa izberemo, ali želimo v sklicu navesti stran, kjer se nahaja element, poglavje, zaporedno številko elementa itd. Ko kliknemo "Vstavi", OpenOffice.org Writer vstavi ustrezen sklic, kar lahko vidimo takoj, če pogovorno okno ne zakriva kazalke. Pogovorno okno zapremo s klikom na tipko "Zapri", ko končamo z vstavljanjem sklicev. Ker je slikc polje, vidimo na zaslonu vrednost na sivi podlagi, ki pa se ne bo odtisnila na papir.

5.2.10 Kazala in stvarna kazala

Knjige in daljši dokumenti so običajno opremljeni s kazalom vsebine, slik in tabel. Strani so ustrezno označene s številkami strani, obsežnejša strokovna besedila pa imajo tudi stvarno kazalo. OpenOffice.org Writer sam številči strani, zato lahko ugotovi, na kateri strani se nahaja nek element besedila, ta podatek pa uporabi pri izgradnji in vzdrževanju ustreznih kazal.

Glava in noga sta dela dokumenta na vrhu in dnu strani, ki lahko vsebujeta podatke, potrebne za hitro identifikacijo vsebine (naslov dela, poglavja, avtor, stran itd).



Slika 5.28: Pogovorno okno za vstavljanje kazala

Dokumentu dodamo glavo preko izbire "Vstavi→Glava→Privzeto" (če v dokumentu uporabljamo več kot en slog stani, kot je to opisano v razdelku 5.2.6, zadnji podmeni ponudi spisek uporabljenih slogov ter možnost Vse). Na vrhu strani se pojavi polje, kamor vnesemo vsebino

glave. Če imamo vklopljen "Pogled→Meje besedila", lahko vidimo, da je glava ločen del dokumenta, ki je, tako kot noga, izven meja besedila.

Nogo dodamo z izbiro "Vstavi→Noga→Privzeto". Vsebina, ki jo vpišemo v glavo in nogo, se bo pojavila na vsaki strani. Glava in noga imata že nastavljeni tabulatorski mesti, na sredini sredinsko poravnano, na desni pa desno poravnano mesto, kar poenostavi vnos vrednosti (o tabulatorskih mestih več v razdelku 5.2.5).

Strani oštevilčimo tako, da v glavo ali nogo dodamo polje "ÂŠtevilka strani" ("Vstavi→Polja→ÂŠtevilka strani").

Dolgim dokumentom ponavadi dodamo vsebinsko in stvarno kazalo. Če želimo, da bo vsebinsko kazalo odražalo dejansko strukturo dokumenta, moramo disciplinirano uporabljati sloge "Naslov"; ti namreč določajo hierarhijo poglavij. "Naslov 1" določa najvišji nivo, "Naslov 2", "Naslov 3" itd, pa nižje nivoje (razdelek 5.2.6).

Kazalo vstavimo z izbiro "Vstavi→Kazala vsebine→Kazala vsebine...". Če želimo izdelati vsebinsko kazalo, v polju "Vrsta" izberemo "Kazalo vsebine". V pogovornem oknu vtipkamo še naslov kazala ter ga primerno oblikujemo. Na podoben način vstavimo tudi kazalo ilustracij, tabel ipd., ki pa bo pravilno nastal le, če smo ustrezne elemente opremili z napisi (razdelek 5.2.9).

Stvarno kazalo vsebuje kazalce na ključne besede, urejene po abecedi. Ključne besede v besedilu označimo tako, da se nanje postavimo s kazalko, nato pa izberemo "Vstavi→Kazala vsebine→Vnos...". V pogovornem oknu izberemo kazalo, v katerem želimo kazalec na besedo, v našem primeru "Abecedno kazalo". Če želimo hierarhično kazalo, določimo tudi nadrejena gesla. Če želimo na primer ključno besedo `Sekvoja` uvrščeno pod `Drevo`, izberemo `Drevo` v polju "1. ključ".

Stvarno kazalo vstavimo na mesto kazalke z "Vstavi→Kazala vsebine→Kazala vsebine" aktiviramo pogovorno okno, v polju "Vrsta" pa nato izberemo "Abecedno kazalo".

5.2.11 Nasveti za učinkovitejše delo

V nadaljevanju bomo obravnavali nekaj napak, ki jih pri sestavljanju dokumentov z besedili delajo začetniki, ter kako odpraviti nekatere težave. Opisane smernice so vredne upoštevanja, saj lahko z njimi dosežemo lažje delo z urejevalnikom besedila, bolj nadzorovano obliko, poleg tega pa bomo lahko naš izdelek učinkoviteje izvozili v kakšno drugo obliko (na primer v HTML).

Podvojeni presledki. Med vnosom in urejanjem besedila včasih pomotoma na mesto, kjer je že presledek, vnesemo še en presledek. Ta dvojni presledek opazimo s pozornim branjem in lahko kvari estetski videz

vrstice. Dvojne presledke med urejanjem lažje odkrijemo, če vklopimo prikaz nenatisljivih znakov ("Pogled→Nenatisljivi znaki" oz. tipka ¶ v glavni orodni vrstici). Odpravimo jih pa tako, da vse nize "␣␣" (dva presledka) v dokumenu zamenjamo z "␣" (en presledek – "Uredi→Najdi in zamenjaj…").

Prazne vrstice in odstavki. V nekaterih dokumentih želimo med odstavki imeti več prostora kot med vrsticami znotraj odstavka. Tudi naslovi in podnaslovi so od predhodnega ali naslednjega odstavka bolj odmaknjeni. Začetniki to dosežejo tako, da na koncu vsakega odstavka ali pred novim naslovom še enkrat pritisnejo `<Enter>` misleč, da tako vstavijo dodatno vrstico, v resnici pa tako dobijo med odstavki prazne odstavke.

Takšen dokument ima potem razdrobljene odstavke, ki jih program drugače razporeja med stranmi, kot to pričakuje uporabnik. Avtorja začetnika tako izdajajo prazni odstavki na začetku nekaterih strani ter osameli naslovi ali podnaslovi na dnu strani.

Veliko boljša praksa je torej, da ne vstavljamo praznih odstavkov pritisnemo `<Enter>` le takrat, ko zares želimo zaključiti odstavek, ki ga vnašamo. Prazen prostor pred in za odstavkom ter razmike med vrsticami nato bistveno bolj nadzorovano nastavimo v lastnostih odstavka (razdelek 5.2.5) ali, še bolje, v lastnostih odstavka posameznega sloga (razdelek 5.2.6).

Tudi prazne odstavke lahko hitro odpravimo tako, da izberemo "Uredi→Najdi in zamenjaj…", v polje "Išči" vnesemo niz "$^" (brez znakov za narekovaje), polje "Zamenjaj z" pustimo prazno in obkljukamo "Regularni izrazi". Nato poženemo iskanje ali zamenjavo s klikom na ustrezno tipko.

Besedilo, razdeljeno med dve strani. V splošnem lahko mesto prehoda na novo stran prepustimo programu, včasih pa želimo, da se določeni deli držijo skupaj na eni strani. Kot primer navedimo odstavek, ki v zadnji povedi z besedilom uvede naštevanje, to pa se začne v naslednjem odstavku. Lahko se zgodi, da se uvajalni odstavek znajde na dnu strani, naštevanje pa se bo začelo na naslednji strani.

Začetniki bi takšen primer ugotovili med pregledovanjem izdelka, težavo pa "rešili" tako, da bi pred uvajalni odstavek vrinili toliko praznih odstavkov, kolikor jih bi bilo potrebno, da problematično besedilo potisnejo na naslednjo stran. Da je to slaba praksa, smo že ugotovili.

Če torej želimo, da nek odstavek preide na novo stran hkrati z naslednjim odstavkom, se s kazalko postavimo na ta odstavek in izberemo "Oblika→Odstavek…", nato pa na strani "Potek besedila" označimo možnost "Ohrani v naslednjem odstavku[1]".

[1] Nekoliko neroden prevod izvirnega "Keep with next paragraph", ustreznejši prevod bi bil "Ohrani skupaj z naslednjim odstavkom".

Velike in male črke. Besedilo običajno pišemo z malimi tiskanimi črkami, velike tiskane črke pa uporabljamo za veliko začetnico, pri kraticah ipd. Uporaba samih velikih tiskanih črk je sicer možna, ni pa najbolj estetska, ker je med drugim ekvivalent zelo glasnega govorjenja.

Če želimo v svojem dokumentu na določenem mestu vseeno uporabiti same velike črke, jih lahko vnesemo tako, da vklopimo `<Caps Lock>` in vnesemo besedilo. Obstoječe besedilo lahko označimo ter preklapljamo med velikimi in malimi črkami z "Oblika→Črke/znaki→Velike črke" in "Oblika→Črke/znaki→Male črke". Slaba stran takšnega urejanja je, da izgubimo informacijo o veliki začetnici. To nam lahko kasneje naredi odvečno delo.

Bolj smiselno je vse besedilo vnesti, kot bi vnesli običajno besedilo z velikimi in malimi tiskanimi črkami. Nato izberemo besedilo in odpremo "Oblika→Znaki…". Na strani "Učinki pisave" v padajočem meniju "Učinki" nastavimo, kako naj program znake prikaže. V vsebini bomo tako ohranili informacijo o veliki in mali začetnici, čeprav razlik na zaslonu ali papirju ne bo.

5.2.12 Naloge

1. Kaj so to proporcionalne pisave? Kakšne pisave glede na širino črk poznamo? Kaj so to serifi?
2. Ali je priporočjivo poljubno povečati ali pomanjšati določeno velikost pisave? Odgovor utemelji!
3. Po vrsti naštejte vsaj sedem delov iz strukture seminarske naloge!
4. Naštejte vsaj pet lastnosti odstavka.
5. Kateri od naslednjih ukazov spremenijo videz izpisa na papirju: vklop prikaza nenatisljivih znakov, sprememba faktorja povečave (zoom), sprememba meja dokumenta, prikaz meja dokumenta?
6. Kakšna je razlika med mehkim in trdim oblikovanjem? Kdaj uporabljamo posamezen primer?
7. Sestavite dokument poljubne vsebine, ki bo imel več naslovov in podnaslovov do nivoja 3. Spremenite lastnosti naslovov nivoja 3. Pred besedilo vstavite kazalo vsebine z naslovom Vsebina, ki prikazuje naslove do nivoja 2.
8. Stran praznega dokumenta napolnite s poljubnim besedilom. Strani nato razdelite na tri stolpce z medsebojnimi razmiki po 5 mm, s črto, ki loči stolpce in ki obsega 75 % višine, poravnano po sredini. Določite še glavo strani, ki naj bo različna na levi in desni strani,

dodatno pa naj jo od besedila loči vodoravna črta (spodnji del obrobe). ÂŠtevilke strani naj bodo na zunanji strani. Odstavke poravnajte na obeh straneh.

9. V dokument iz predhodne naloge vstavite sliko iz galerije ali datoteke. Sliko zmanjšajte ali povečajte tako, da bo nekoliko ožja od širine stolpca, in jo postavite med dva stolpca. Sliki dodajte napis, nastalemu okviru pa določite vzporedno oblivanje besedila (oblivanje strani). Da se bo okvir laže ločil od besedila, mu določite tanko obrobo ter povečajte odmik od besedila.

Zadeva:·Vabilo·na·i·z·l·e·t¶
¶
Vabimo·vas·na·izlet·v·neznano,·ki·bo
naslednjo·sredo.·Prijave·pobiramo·do·petka.
Program·izleta:¶
¶
··7:30·····zbor·pred·stavbo¶
··7:45·····preštevanje·zbranih¶
··7:50·····izbira·vodje·izleta¶
··8:00·····oskrba·morebitnih·ranjencev¶
··8:15·····odhod·v·neznano·smer¶
·20:00·····povratek·domov¶
¶
¶
·····tajnik¶
·····Tipko·Počasnik¶

Slika 5.29: Izdelek začetnika

10. Od začetnika prejmemo dokument, ki po vklopu prikaza nenatisljivih znakov na zaslonu izgleda tako, kot prikazuje slika 5.29. Popravite dokument tako, da bodo presledki samo še enojni in še ti le med dvema besedama. Lahko odpravimo tudi prazne odstavke?

11. Oblikujte naslovno stran seminarske naloge. V zgornjem delu strani naj bo ime univerze in fakultete, v osrednjem delu naslov in podnaslov (npr. "seminarska naloga pri predmetu. . . "), pod naslovom imena avtorjev, na dnu strani, pa vstavite polje za datum. Dodate lahko tudi logotip univerze, če ga najdete na internetu. Pri oblikovanju ne uporabljajte praznih odstavkov in dodatnih presledkov![2]

[2]Namig: besedilo kot funkcija risanja, ki ga zasidrate na stran.

5.3 LaTeX

5.3.1 Uvod

LaTeX je sistem za logično urejanje besedil. TeX, ki je osnova sistema LaTeX, je leta 1977 začel razvijati Donald Knuth z Univerze Stanford v ZDA [8]. Profesor Knuth je napisal veliko knjig s področja računalništva. Ker ni imel na voljo dovolj dobrega orodja za pisanje besedil, ki bi omogočal med drugim tudi pisanje zapletenih matematičnih izrazov, je sklenil, da ga napiše kar sam. Tako je nastal TeX, sistem za oblikovanje besedil, ki je posebej primeren za pisanje znanstvenih besedil z veliko matematike. TeX [7] je hitro postal priljubljen tudi pri drugih znanstvenikih in na drugih znanstvenih področjih ter pri vseh tistih, ki želijo zelo kvalitetno in fleksibilno tipografsko oblikovanje. Danes je TeX standard na številnih založniških področjih. Uporablja se za tehnične in naravoslovne knjige, konferenčne zbornike, slovarje in leksikone, večjezične knjige itd. Implementacije sistema TeX obstajajo praktično za vse obstoječe operacijske sisteme. Zaradi stabilnosti in kompatibilnosti se je Donald Knuth odločil, da se TeX ne bo več spreminjal.

V začetku 80. let je Leslie Lamport začel razvijati LaTeX [9], ki je sistem makro ukazov na osnovi sistema TeX. Kot vsak sistem oziroma programska oprema se je tudi LaTeX razvijal skozi več različic. Trenutno je v uporabi različica LaTeX2ϵ, ki jo obravnavamo tudi v tej knjigi. TeX in LaTeX sta v javni rabi in zato obstaja cela vrsta njunih brezplačnih implementacij, ki so dosegljive na svetovnem spletu.

Osnovne datoteke in način dela

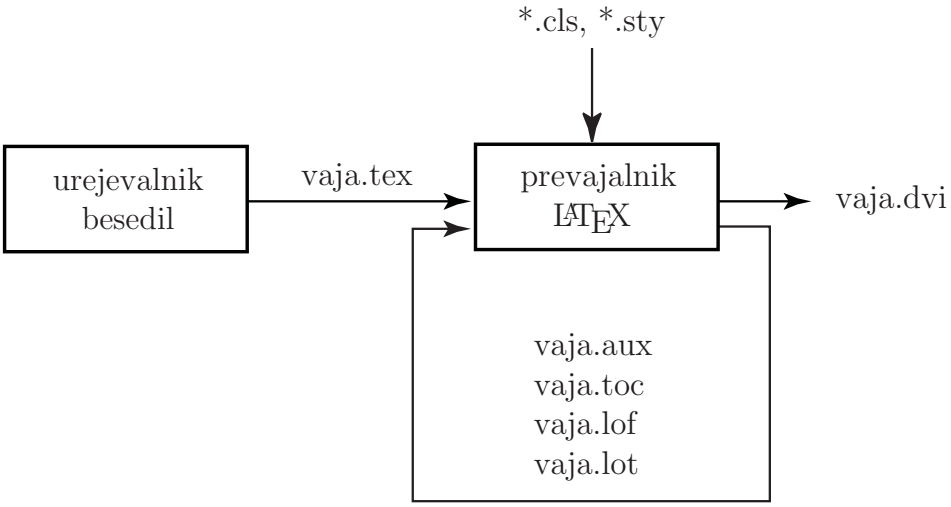
V urejevalniku besedil pišemo besedilo, ki ga opremimo z ukazi za formatiranje. To z ukazi za formatiranje obogateno besedilo moramo nato prevesti, da dobimo formatirano besedilo za izpis na tiskalniku (slika 5.30). Prevajalnik pri prevajanju upošteva tudi razne stilske datoteke, ki definirajo obliko formatiranega besedila. Pri prevajanju se poleg formatiranega besedila generira še cela vrsta pomožnih datotek, ki jih prevajalnik uporabi pri ponovnem prevajanju.

Vhodno datoteko v sistemu LaTeX, ki jo lahko pišemo s poljubnim urejevalnikom, označimo s podaljškom `tex`: na primer `vaja.tex`. Oglejmo si podaljške in vlogo najpomembnejših datotek:

`*.tex` vhodna datoteka z besedilom in ukazi za formatiranje,

`*.dvi` formatirana izhodna datoteka oziroma prevod datoteke `*.tex`,

`*.aux` pomožna datoteka,



Slika 5.30: Način dela s sistemom LaTeX in osnovne datoteke

`*.toc` kazalo,

`*.lof` seznam slik,

`*.lot` seznam tabel,

`*.sty` stilske datoteke,

`*.cls` oblikovne predloge.

Pri obsežnejših besedilih si pri navajanju literature lahko pomagamo tudi z datotekami BIBTeX, v katere shranjujemo podatke o citirani literaturi (podaljška `bbl` in `bib`).

5.3.2 Značilnosti sistema LaTeX pod Linuxom

Pisanje kode LaTeX v urejevalniku

Ena izmed tipičnih napak začetnih uporabnikov sistema LaTeX, ki smo jo spoznali skozi prakso, je ta, da želijo kodo zapisati kar v urejevalniku OpenOffice.org Writer. Ta urejevalnik v osnovi seveda ne shranjuje datotek v tekstovni obliki, zato pozorno preberite naslednji odstavek, ki opisuje pravilen postopek pisanja kode!

Pri opisu dela v terminalskem načinu Linuxa smo spoznali ukaz `emacs`, s katerim zaženemo istoimenski urejevalnik! V primeru, da na priloženi zgoščenki Slix tega ukaza ne pozna, poskusite z ukazom `xemacs` ali pa z ukazom `Emacs`. Če želimo ustvariti novo datoteko z imenom `vaja.tex`, v ukazni vrstici napišemo `emacs vaja.tex &` in ukaz potrdimo s pritiskom

na tipko `<Enter>`. Odpre se nam okno urejevalnika, ki vsebuje vsebino zahtevane datoteke. Seveda, če ta datoteka še ne obstaja, je vsebina okna prazna. Uporabljamo seveda lahko tudi katerega od preprostejših urejevalnikov besedil.

Obstajajo tudi razna integrirana orodja, nekakšni vmesniki, ki omogočajo večjo preglednost nad sintakso kode s pomočjo barv, lažji dostop do prevajalnika in prikazovalnika, do osnovnih gradnikov ipd. Na Slixu se takšno orodje imenuje `kile`, `texmacs` pa je sorodno orodje, ki omogoča urejanje po načelu WYSIWYG.

Prevajanje kode LaTeX

Ko končamo z delom v urejevalniku `emacs`, kar pomeni, da smo shranili vsebino v datoteko s končnico `tex`, moramo ustvarjeno datoteko prevesti (slika 5.30). Če želimo prevesti datoteko z imenom `vaja.tex`, moramo v ukazni vrstici napisati:

```
latex vaja.tex
```

Če v kodi ni napak, bo LaTeX javil, da je uspešno ustvaril prevod, sicer pa izpostavi mesto, ki ga ne more prevesti in poskuša identificirati napako. Ta informacija skoraj vedno že zadostuje za odpravo napake.

Prikaz prevoda

Ko odpravimo vse napake v kodi in uspešno prevedemo datoteko, moramo le še prikazati prevod na ekranu. To storimo s klicem prikazovalnika datotek s končnico `dvi`, na primer z ukazom `xdvi`. Če želimo prikazati prevod ustvarjene datoteke z imenom `vaja.tex`, moramo v ukazni vrstici napisati:

`xdvi vaja` ali `xdvi vaja.dvi`

Recimo, da želimo sedaj prevod, ki ima podaljšek `dvi`, spremeniti v datoteko PostScript, ki ima podaljšek `ps`. To naredimo tako, da v ukazni vrstici uporabimo ukaz `dvips`:

```
dvips -o vaja.ps vaja.dvi
```

Datoteko PostScript lahko sedaj pošljemo direktno na tiskalnik PostScript z ukazom `lpr vaja.ps`.

Podobno lahko z ukazom `pdflatex` ustvarimo tudi datoteko, ki ima podaljšek `pdf`:

```
pdflatex vaja.tex
```

Na izhodu dobimo datoteko `vaja.pdf`. Format PDF (Portable Document Format) podjetja Adobe Systems Incorporated postaja standarden format za prenos elektronskih dokumentov po svetovnem spletu.

5.3.3 Zgradba datotek s končnico `tex`

Vhodna datoteka sistema LaTeX je sestavljena iz glave, jedra in repa. Jedro predstavlja vsebino dokumenta, zato si oglejmo, kako izgledata glava in rep datoteke s končnico `tex`:

```
\documentclass[12pt,a4paper]{article}
\usepackage[slovene]{babel}
\begin{document}

\end{document}
```

Prve tri vrstice bodo za nas predstavljale glavo dokumenta, zadnja pa rep. Med glavo in rep vnesemo vsebino dokumenta.

Tukaj pa velja omeniti naslednjo tipično napako začetnih uporabnikov sistema LaTeX: datoteko pozabijo opremiti z glavo in repom. Takšna datoteka pa se seveda ne more prevesti uspešno. Zato v datoteki vedno najprej ustvarite glavo in rep, nato pa dodajte med ta dva dela željeno vsebino!

Na kratko komentirajmo osnovne lastnosti sistema LaTeX, ki so razvidne iz zapisane kode:

- Vsebina dokumenta mora vedno biti vpeta med glavo in rep dokumenta, sicer se dokument ne bo prevedel.
- Vsak ukaz se v sistemu LaTeX začne z znakom `\`.
- Oznaka `12pt` podaja v pikah osnovno velikost pisave v dokumentu. Prav tako bi lahko na primer uporabili oznako `11pt` za pisavo velikosti 11 pik. Če ne predpišemo velikosti, LaTeX samodejno uporabi pisavo velikosti 10 pik.
- Oznaka `a4paper` podaja velikost papirja, ki ga uporabljamo za natis dokumenta.
- Oznaka `article` podaja vrsto dokumenta. LaTeX pozna več vrst dokumentov, ki pravzaprav predstavljajo oblikovne vzorce, največkrat uporabljene pa so:

  `article` – oblikovna predloga za pisanje člankov,

`report` – oblikovna predloga za pisanje poročil,

`book` – oblikovna predloga za pisanje knjig.

- V drugi vrstici glave sistemu LaTeX povemo, da bomo uporabljali paket `babel` (Babilon po slovensko). Z izbiro jezika `slovene` zahtevamo, da se vse pomožne besede, ki se samodejno generirajo v dokumentu, izpišejo v slovenščini namesto v angleščini: *contents/kazalo*, *chapter/poglavje*, *bibliography/literatura*, *figure/slika* itd. Paket `babel` vsebuje opcije za skoraj vse jezike, ki uporabljajo latinsko pisavo.

- Osnovni gradniki datotek sistema LaTeX so okolja, ki imajo skupno osnovno sintakso:

```
\begin{xyz}
\end{xyz}
```

pri čemer `xyz` ponazarja ime nekega okolja: `document`, `figure`, `table`, `array`, `eqarray`, `tabular`, `verbatim`, `itemize`, `enumerate`, `equation`, `thebibliograhy` itd. Med `begin` in `end` postavimo ukaze, bolj ali manj specifične za določeno okolje. Okolja lahko tudi gnezdimo. Več o okoljih bomo spregovorili v nadaljevanju. Na tem mestu pa izpostavimo zgolj dejstvo, da je okolje vseh okolij okolje `document`:

```
\begin{document}
\end{document}
```

5.3.4 Kako pišemo šumnike?

Šumniki sicer niso edini posebni znaki, ki jih angleščina ne pozna, vendar si zaradi njihovega pomena pri pisanju slovenščine najprej oglejmo pisanje šumnikov. ÂŠumnike lahko pišemo na več načinov:

- Z ukazom `\v{c}` dobimo č.

- Z ukazom `\v c` prav tako dobimo č.

- Če v glavi dokumenta za prvo vrstico dodamo naslednjo kodo:

```
\catcode`\''=13
\def''#1{\v #1}
```

ali če uporabljamo paket `babel` z opcijo `slovene`:

```
\usepackage[slovene]{babel}
```

lahko č zapišemo še krajše in bolj pregledno kot `"c`[3].

- Če pišemo besedilo v kodnem naboru ISO-8859-2 (Latin 2) (ta je v slovenskem okolju navadno privzeti nabor), potem lahko v glavo dokumenta za prvo vrstico dodamo ukaz:

  ```
  \usepackage[latin2]{inputenc}
  ```

  Ta nam omogoča, da lahko šumnike pišemo kar direktno v urejevalnik in jih po prikazu prevoda na zaslonu tudi pravilno vidimo.

  Podobno, če uporabljamo kodni nabor Windows-1250 za pisanje šumnikov, uporabimo ukaz:

  ```
  \usepackage[cp1250]{inputenc}
  ```

5.3.5 Posebni znaki

Podobno kot šumnike lahko zapišemo tudi črke drugih jezikov in naglase. Oglejmo si nekaj najpogostejših:

ć	`\'{c}`	ê	`\^{e}`
è	`\`{e}`	ë	`\"{e}`
ç	`\c{c}`	ł	`\l`
å	`\aa`	ß	`\ss`
£	`\pounds`		

V sistemu LaTeX imamo deset znakov s posebnim pomenom. Pomen znaka \ smo že spoznali, ostali pa so:

```
# $ % & ~ _ ^ { }
```

Pomen posameznih posebnih znakov bomo spoznali v nadaljevanju. Včasih je potrebno te znake tudi izpisati. To storimo na naslednji način:

#	`\#`	&	`\&`
_	`\_`	$	`\$`
%	`\%`	{	`\{`
}	`\}`	\	`$\backslash$`
^	`$^\wedge$`	~	`$^\sim$`

V zadnjih treh primerih je ukaz znotraj matematičnega okolja, to je med znakoma $. Več o matematičnih okoljih bomo povedali v nadaljevanju.

[3] V kodi in tukaj uporabljamo dvojni narekovaj.

5.3.6 Slogi in velikosti pisav

V sistem LaTeX je vključena družina pisav *Computer Modern*, ki jo je oblikoval avtor TeX-a Donald Knuth. V sistemu LaTeX pa seveda lahko uporabljamo tudi druge pisave, predvsem je enostavna uporaba pisav v formatu PostScript. Ta učbenik je napisan s pomočjo sistema LaTeX in z družino pisav Computer Modern.

V sistemu TeX lahko stavimo tudi besedila, ki ne uporabljajo latinske pisave, na primer v cirilici, grščini, arabščini, hebrejščini, japonščini in še celi vrsti drugih pisav, ki se ne pišejo le z leve proti desni ali od zgoraj navzdol kot latinska pisava.

Najprej si oglejmo ukaze za osnovne sloge pisav:

`\rm`	pokončno – `{\rm pokon\v cno}`
`\it`	*kurziva* – `{\it kurziva}`
`\sl`	*elektronska kurziva* – `{\sl elektronska kurziva}`
`\bf`	**krepko** – `{\bf krepko}`
`\sc`	MALE KAPITELKE – `{\sc male kapitelke}`
`\sf`	linearna pisava – `{\sf linearna pisava}`
`\tt`	`pisalni stroj` – `{\tt pisalni stroj}`

Privzeti slog pisave v sistemu LaTeX je pokončni (roman `\rm`). Kot je razvidno iz zgornjih primerov, delovanje omenjenih ukazov omejimo tako, da jih vstavimo med zavita oklepaja {}.

Pomanjkljivost zgornjih ukazov je, da jih ni možno kombinirati, zato zapišimo še primere ekvivalentnih ukazov, ki omogočajo tudi kombiniranje slogov pisav:

`\textrm`	pokončno – `\textrm{pokon\v cno}`
`\textit`	*kurziva* – `\textit{kurziva}`
`\textsl`	*elektronska kurziva* - `\textsl{elektronska kurziva}`
`\textbf`	**krepko** – `\textbf{krepko}`
`\textsc`	MALE KAPITELKE – `\textsc{male Kapitelke}`
`\textsf`	linearna pisava – `\textsf{linearna pisava}`
`\texttt`	`pisalni stroj` – `\texttt{pisalni stroj}`

Sedaj lahko zapišemo tudi poljubne kombinacije zgornjih slogov pisav:

Pomembno pa je,	`\textsf{\textbf{Pomembno pa je,}}`
DA LAHKO PIŠEMO	`\texttt{\textsc{da lahko pi\v semo}}`
v takšnem ali	`\textsf{\textit{v tak\v snem ali}}`
drugačnem slogu.	`\texttt{\textit{druga\v cnem slogu.}}`

Osnovna velikost pisave je, kot že rečeno, podana v glavi datoteke s končnico `tex`, vendar jo lahko znotraj dokumenta spreminjamo z naslednjimi relativnimi ukazi:

`\tiny`	beseda – `{\tiny beseda}`
`\scriptsize`	beseda – `{\scriptsize beseda}`
`\footnotesize`	beseda – `{\footnotesize beseda}`
`\small`	beseda – `{\small beseda}`
`\normalsize`	beseda – `{\normalsize beseda}`
`\large`	beseda – `{\large beseda}`
`\Large`	beseda – `{\Large beseda}`
`\LARGE`	beseda – `{\LARGE beseda}`
`\huge`	beseda – `{\huge beseda}`
`\Huge`	beseda – `{\Huge beseda}`

5.3.7 Struktura dokumenta

Omenili smo že priporočljivo strukturo seminarskih nalog, v tem poglavju pa si oglejmo, kako s sistemom LaTeX ustvarimo naslovno stran, poglavja, dvokolonski dokument, spisek uporabljene literature, kako se sklicujemo na literaturo in kako ustvarimo kazalo, spisek slik in tabel.

Vzemimo pod drobnogled besedilo v sistemu LaTeX na sliki 5.31. Njegov prevod je prikazan na sliki 5.32. Komentirajmo neznane ukaze:

- S posebnim znakom `%` lahko zapišemo komentarje v dokumentu. Ti komentarji v prevodu niso vidni.
- Če za ukazom `\begin{document}` uporabimo ukaze `\title`, `\author` in `\date`, kjer med zavitima oklepajema {} navedemo naslov, avtorje in datum, nato pa za temi ukazi uporabimo ukaz `\maketitle`, bo LaTeX ob prevodu ustvaril naslov, če je izbrani tip dokumenta `article`, ali naslovno stran, če je izbrani tip dokumenta `book` ali `report`.
- Poglavja in razdelke ustvarjamo z ukazi: `\section` (razdelek), `\subsection` (podrazdelek), `\subsubsection` (pod-podrazdelek) in `\chapter` (glavno poglavje, ki ga lahko uporabimo le v tipu dokumenta `book` ali `report`). Ime poglavja navedemo med zavitima oklepajema {}, ki sledijo ukazu.
- O okolju `\thebibliography` bomo spregovorili malo kasneje.

In kako ustvarimo dvokolonski dokument? Enostavno tako, da prvo vrstico glave dokumenta iz

```
\documentclass[12pt,a4paper]{article} %Ta dokument bo enokolonski!
\usepackage[slovene]{babel}

\begin{document}

\title{Superkvadri\v cni modeli}
\author{Franc Solina}
\date{6. julij 2000}
\maketitle

\section{Definicija}
Superkvadriki so 3D modeli, ki se uporabljajo v ra\v cunalni\v skem
vidu za mo\-deliranje in segmentacijo globinskih slik
\cite{Kluwer_2000}.

\subsection{Razvoj metode}
Rekonstrukcijo posami\v cnih superkvadrikov~\cite{london_87} smo
zdru\v zili z metodo segmentacije ``opi\v si in izberi''
\cite{leonardis93}. Rekonstrukcijo in segmentacijo superkvadrikov
smo testirali na globinskih slikah~\cite{PAMI97}.

\begin{thebibliography}{1}
\bibitem{london_87}
R. Bajcsy and F. Solina. Three dimensional object representation
revisited. In {\em Proceedings First International Conference on
Computer Vision}, pages 231--240, London, England, June 1987.
\bibitem{Kluwer_2000}
A. Jakli\v{c}, A. Leonardis, and F. Solina. {\em Segmentation and
Recovery of Superquadrics}. Kluwer, Dordrecht, 2000.
\bibitem{leonardis93}
A. Leonardis. {\em Image Analysis Using Parametric Models:
Model-Recovery and Model-Selection Paradigm}. PhD thesis,
University of Ljubljana, Faculty of Electrical Engineering and
Computer Science, Ljubljana, Slovenia, 1993.
\bibitem{PAMI97}
A. Leonardis, A. Jakli\v{c}, and F. Solina. Superquadrics for
segmentation and modeling range data. {\em IEEE Transactions on
Pattern Recognition and Machine Intelligence}, 19(11):1289--1295,
November 1997.
\end{thebibliography}

\end{document}
```

Slika 5.31: Izvorno besedilo v sistemu LaTeX

Superkvadrični modeli

Franc Solina

6. julij 2000

1 Definicija

Superkvadriki so 3D modeli, ki se uporabljajo v računalniškem vidu za modeliranje in segmentacijo globinskih slik [2].

1.1 Razvoj metode

Rekonstrukcijo posamičnih superkvadrikov [1] smo združili z metodo segmentacije "opiši in izberi" [3]. Rekonstrukcijo in segmentacijo superkvadrikov smo testirali na globinskih slikah [4].

Literatura

[1] R. Bajcsy and F. Solina. Three dimensional object representation revisited. In *Proceedings First International Conference on Computer Vision*, pages 231–240, London, England, June 1987.

[2] A. Jaklič, A. Leonardis, and F. Solina. *Segmentation and Recovery of Superquadrics*. Kluwer, Dordrecht, 2000.

[3] A. Leonardis. *Image Analysis Using Parametric Models: Model-Recovery and Model-Selection Paradigm*. PhD thesis, University of Ljubljana, Faculty of Electrical Engineering and Computer Science, Ljubljana, Slovenia, 1993.

[4] A. Leonardis, A. Jaklič, and F. Solina. Superquadrics for segmentation and modeling range data. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 19(11):1289–1295, November 1997.

Slika 5.32: Prevod kode s slike 5.31

Superkvadrični modeli

Franc Solina

6. julij 2000

1 Definicija

Superkvadriki so 3D modeli, ki se uporabljajo v računalniškem vidu za modeliranje in segmentacijo globinskih slik [2].

1.1 Razvoj metode

Rekonstrukcijo posamičnih superkvadrikov [1] smo združili z metodo segmentacije "opiši in izberi" [3]. Rekonstrukcijo in segmentacijo superkvadrikov smo testirali na globinskih slikah [4].

Literatura

[1] R. Bajcsy and F. Solina. Three dimensional object representation revisited. In *Proceedings First International Conference on Computer Vision*, pages 231–240, London, England, June 1987.

[2] A. Jaklič, A. Leonardis, and F. Solina. *Segmentation and Recovery of Superquadrics*. Kluwer, Dordrecht, 2000.

[3] A. Leonardis. *Image Analysis Using Parametric Models: Model-Recovery and Model-Selection Paradigm*. PhD thesis, University of Ljubljana, Faculty of Electrical Engineering and Computer Science, Ljubljana, Slovenia, 1993.

[4] A. Leonardis, A. Jaklič, and F. Solina. Superquadrics for segmentation and modeling range data. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 19(11):1289–1295, November 1997.

Slika 5.33: Dvokolonska postavitev kode s slike 5.31

```
\documentclass[12pt]{article}
```

spremenimo v

```
\documentclass[12pt,twocolumn]{article}
```

Rezultat te zamenjave v izvornem besedilu je viden na sliki 5.33.

Komentirajmo še osnovne gradnike v poglavju **Literatura** s slike 5.32:

- Vire naštejemo v okviru okolja `thebibliography`.
- Z oznako `{1}` povemo sistemu LaTeX, da bomo navedli največ 9 virov. Z oznako `{11}` pa bi povedali, da bo v spisku literature največ 99 virov. S pomočjo te informacije LaTeX ustrezno vertikalno poravna oštevilčenje virov.
- Ukaz `\bibitem` uporabimo za navajanje posameznih virov. Med zavitima oklepajema {} nato podamo simbolično ime vira, ki ga bomo v dokumentu uporabili pri sklicevanju na ta vir, nadaljujemo pa z opisom vira.
- Največkrat ime vira poimenujemo po prvem avtorju vira, pri tem pa moramo seveda biti pozorni, da se nam imena ne ponavljajo. V našem primeru smo prvi vir poimenovali po kraju konference (`london_87`), drugega po založbi (`Kluwer_2000`), tretji vir pa po avtorju (`leonardis93`). Razlikovati moramo med velikimi in malimi črkami!

Ko ustrezno poimenujemo vire, se lahko nanje v besedilu tudi sklicujemo, LaTeX pa poskrbi, da se številčenje referenc samodejno obnavlja. Za pravilno številčenje je po vsaki spremembi v literaturi potrebno izvorno kodo LaTeX **vedno prevesti dvakrat**. V izvornem besedilu se na posamezen vir sklicujemo s pomočjo ukaza `cite`, kar je razvidno iz primera na sliki 5.31. Kot privzet način prikazovanja citiranih virov je v sistemu LaTeX uporabljeno številčenje v oglatih oklepajih. Z uporabo drugih slogov citiranja je možno citate prikazati tudi na druge načine, na primer s priimki avtorjev in letnico izdaje v okroglih oklepajih – (Bajcsy and Solina, 1991).

Omenjeni način podajanja virov, ko kar v okviru izvornega besedila naštejemo vse vire, je smotrn, če je virov malo in če istih virov ne uporabljamo večkrat. Sicer pa se nam izplača vzdrževati lastno bazo virov (datoteke s podaljškom `bib`), iz katere s pomočjo programa BIBTeX samodejno generiramo spisek citirane literature. To so datoteke s podaljškom `bbl`, ki jih LaTeX vključi namesto okolja `thebibliography`.

V našem primeru smo v spisku literature na slikah 5.31 in 5.32 uporabili:

1. članek, objavljen v konferenčnem zborniku,
2. knjigo
3. doktorsko disertacijo in
4. članek, objavljen v znanstveni reviji.

To so štiri najpogostejše vrste publikacij, ki jih citiramo v strokovnih in znanstvenih besedilih. Pozorno poglejte, katere elemente in po kakšnem vrstnem redu jih je potrebno navesti v spisku literature za posamezno vrsto publikacije.

Zaradi logičnega urejanja besedil nam LaTeX omogoča enostavno ustvarjanje kazal ter spiskov slik in tabel. Na mestu, kjer v dokumentu želimo, da se pojavi določen seznam, enostavno zapišemo ustrezen ukaz:

`\tableofcontents`	ustvari kazalo
`\listoffigures`	ustvari spisek slik
`\listoftables`	ustvari spisek tabel

prevedemo dvakrat in dobimo rezultat.

5.3.8 Okolja

Do sedaj smo spoznali že dve osnovni okolji: `document` in `thebibliography`. V tem poglavju bomo omenili še ostala pomembna okolja, z izjemo tistih, ki so specifična za pisanje matematičnih izrazov.

Sredinska poravnava

Če želimo nek del dokumenta poravnati na sredino strani oziroma kolone, uporabimo okolje `center`:

```
\begin{center}
\end{center}
```

Kot smo že omenili, želeni del dokumenta damo med obe vrstici, torej med `begin` in `end`. To velja za vsa okolja tipa `begin-end`.

Naštevanje

Okolje za naštevanje se imenuje `itemize`:

```
\begin{itemize}
\item
\end{itemize}
```

Vsako točko spiska naredimo z ukazom `\item`, ki mu sledi želeni del besedila. Na primer: `\item Potrdilo o dr\v zavljanstvu`.

ÂŠtevilčenje

Okolje za številčenje se imenuje `enumerate`:

```
\begin{enumerate}
\item
\end{enumerate}
```

Tudi tukaj naredimo točko spiska z ukazom `\item`, le da jih v tem primeru LaTeX oštevilči.

Opisno naštevanje

Okolje za opisno naštevanje se imenuje `description` in nam omogoča, da namesto začetnega znaka ali številke spiska uporabimo (skoraj) poljubno zaporedje:

```
\begin{description}
\item[]
\end{description}
```

Podobno kot pri prejšnjih dveh okoljih, tudi tukaj naredimo točko spiska z ukazom `\item`. ÂŽeleno zaporedje, ki ga bomo uporabili namesto znaka ali številke spiska, zapišemo med oglate oklepaje `[]`. Na primer:

```
\item[Potrdila:] o dr\v zavljanstvu,\ldots  .
```

Okolje za dobesedni izpis

Okolje `verbatim` nam omogoča, da izpišemo besedilo točno tako, kot smo ga zapisali v vhodni datoteki s končnico `tex`:

```
\begin{verbatim}
\end{verbatim}
```

To okolje nam pride prav predvsem tam, kjer želimo izpostaviti strukturo nekega dela dokumenta (koda programa, ukaz LaTeX itd.). LaTeX bo prikazal vsak presledek, vsak znak, vsako število in vsak razlom vrstice točno tako, kot smo zapisali. Primer:

```
\begin{verbatim}
For i:= 1 To 20 Do
  Write('\v{c}\v{s}\v{z}')
\end{verbatim}
```

bomo v prevodu videli kot:

```
For i:= 1 To 20 Do
  Write('\v{c}\v{s}\v{z}')
```

Vrstični dobesedni izpis

Okolje za vrstični dobesedni izpis se pomensko razlikuje od prej opisanega okolja le po tem, da lahko prikaže le nekaj besed ali največ eno vrstico besedila, saj tega okolja LaTeX ne razdeli na več vrstic, tudi če je okolje daljše od možne dolžine vrstice:

```
\verb++
```

Največkrat nam to okolje pride prav tam, kjer moramo med samim besedilom dokumenta izpisati besedilo točno tako, kot smo ga zapisali v vhodni datoteki s končnico `tex`. Ukaz, ki ga uporabimo, se imenuje `\verb`; besedilo, ki ga izpisujemo, pa vkleščimo med dva enaka znaka, na primer +. Primer:

```
\verb+__Write('\v{c}\v{s}\v{z}')+
```

bomo v prevodu videli kot: `__Write('\v{c}\v{s}\v{z}')`.

Pisanje opomb

Če želimo v dokumentu zapisati opombo, ki se bo pojavila na dnu strani, se postavimo na tisto mesto v dokumentu, na katero se opomba nanaša, in zapišemo opombo s pomočjo ukaza `\footnote`, kjer med zavita oklepaja {} zapišemo želeno opombo:

```
\footnote{}
```

Okolje za poravnavo

Okolje za poravnavo `tabular` je pravzaprav osnovno okolje za gradnjo tabel, ki si ga bomo ogledali v naslednjem poglavju. Razložimo delovanje na naslednjem primeru:

```
\begin{tabular}{|clr|}\hline
ena & dva & tri\\\cline{2-2}
1 & 2 & 3\\\hline
\end{tabular}
```

V prevodu ta del kode vidimo kot:

ena	dva	tri
1	2	3

Komentar kode:

- Osnovno okolje:

  ```
  \begin{tabular}
  \end{tabular}
  ```

- Pomen izraza med zavitima oklepajema `{|clr|}`:
  - znak `c` podaja ukaz sistemu LaTeX, da naj bo besedilo v prvem stolpcu sredinjeno (*c – center*),
  - znak `l` podaja ukaz sistemu LaTeX, da naj bo besedilo v drugem stolpcu poravnano levo (*l – left*),
  - znak `r` podaja ukaz sistemu LaTeX, da naj bo besedilo v tretjem stolpcu poravnano desno (*r – right*),
  - znak `|` podaja ukaz sistemu LaTeX, da naj bo na podanem mestu v tabeli navpična črta,

  - celoten izraz pove, da smo ustvarili strukturo s tremi stolpci, ki je na začetku in na koncu omejena z navpično črto. Skupno število znakov `c`, `l` in `r` torej določa število stolpcev v tabeli.

- Ukaz `\hline` naredi vodoravno črto.
- Ukaz `&` določa mejo med stolpci.
- Ukaz `\cline{2-2}` naredi vodoravno črto čez drugi stolpec. Če bi namesto tega napisali `\cline{2-3}`, bi LaTeX naredil črto čez drugi in tretji stolpec.
- Ukaz `\\` določa zaključek vrstice.

Na tem mestu omenimo le še ukaz za združevanje stolpcev:

```
\multicolumn{}{}{}
```

Primer:

```
\begin{tabular}{|clr|}\hline
\multicolumn{2}{||c||}{1+2} & 3\\\hline
ena & dva & tri\\\cline{2-2}
1 & 2 & 3\\\hline
\end{tabular}
```

V prevodu ta del kode vidimo kot:

1+2		3
ena	dva	tri
1	2	3

Komentar kode v ukazu `\multicolumn`:

- Med prvim parom zavitih oklepajev {} podamo število, ki pove, koliko stolpcev združujemo.
- Med drugim parom zavitih oklepajev {} podamo izgled novega, združenega stolpca. V našem primeru ga sredinimo, na levem in desnem robu pa ga omejimo z dvema navpičnima črtama.
- Med tretjim parom zavitih oklepajev {} zapišemo besedilo, ki naj se izpiše v združenem stolpcu.

Tabele

Za gradnjo tabel uporabimo okolje `table`, znotraj njega pa že omenjeno okolje `tabular`:

```
\begin{table}[htb]

\begin{tabular}
\ldots
\end{tabular}

\caption{}
\label{}
\end{table}
```

Okolje `table` je plavajoče okolje, kar pomeni, da LaTeX sam smiselno uvrsti tabelo po prevodu dokumenta in jo oštevilči. Če bi uporabili zgolj okolje `tabular`, pa bi se ta struktura pojavila točno na tistem mestu v besedilu, kjer smo jo zapisali. Problem pri razporejanju nastopi seveda takrat, ko na strani ni več dovolj prostora za neko tabelo in zato na tistem mestu ostane prazen prostor. Te težave reši plavajoče okolje `table`, ki mu lahko še namignemo, kam naj tabelo postavi. Opcije v oglatih oklepajih `[ ]`, ki sledijo ukazu `\begin{table}`, pomenijo:

- S črko `h` sistemu LaTeX namignemo, da želimo imeti tabelo točno na tem mestu (*h – here*).
- S črko `t` sistemu LaTeX namignemo, da želimo imeti tabelo na vrhu strani (*t – top*).
- S črko `b` sistemu LaTeX namignemo, da želimo imeti tabelo na dnu strani (*b – bottom*).
- S črko `p` sistemu LaTeX namignemo, da naj ustvari posebno stran, na kateri naj bodo le plavajoča okolja (tabele in slike) (*p – page*).
- Možne so seveda tudi smiselne kombinacije. Kombinacija `htb` pomeni, da naj najprej poskuša tabelo prikazati točno na tem mestu, nato na vrhu strani, sicer pa ob njenem vznožju.

Ukaz `\caption` nam omogoča, da tabeli dodamo spremno besedilo, ki ga vpišemo med zavita oklepaja {}. ÂŠtevilčenja ni potrebno zapisovati, saj za to poskrbi LaTeX sam.

Tudi tabele lahko poimenujemo, kar nam nato omogoča, da se nanje sklicujemo v besedilu, podobno kot na vire iz poglavja o literaturi. Ime predpišemo z ukazom `\label`, kjer dejansko ime zapišemo med zavita oklepaja {}:

```
\label{rezultati}
```

V besedilu se na tabelo z imenom `rezultati` sklicujemo z ukazom `\ref`:

```
\ref{rezultati}
```

Da bo ustrezen sklic viden v prevedenem dokumentu, moramo seveda tudi tu dvakrat prevesti naš dokument. Ker sklic predstavlja zgolj ustrezno številko, ga moramo navadno opremiti tudi z ustreznim kontekstom. Oglejmo si primer kode in prevoda:

```
V tabeli~\ref{rezultati} lahko\ldots
```

V tabeli 1 lahko...

Z znakom ~ smo sistemu LaTeX ukazali, da naj prevod besedila `tabeli~\ref{rezultati}` prikaže v isti vrstici, torej naj ga na tem mestu ne razdeli na dve vrstici. Ustrezen kontekst k številki tabele v zgornjem primeru je beseda `tabeli`.

Podobno velja tudi za sklicevanje na poglavja, razdelke, slike in enačbe. Pri dodeljevanju imen moramo biti pozorni, da se nam imena ne ponavljajo.

Vključevanje slik

Čeprav je v sistemu LaTeX možno izdelati preproste diagrame, večino slikovnega gradiva izdelamo z drugimi grafičnimi orodji. Vse slike v formatu PostScript lahko enostavno vključimo v dokumente LaTeX s pomočjo paketov `graphics` in `epsfig`.

Okolje `figure` je v osnovi zelo podobno okolju `table`:

```
\begin{figure}[htb]

\psfig{figure=,width=}

\caption{}
\label{}
\end{figure}
```

zato omenimo le bistvene razlike in zahteve:

- V tretjo vrstico glave dokumenta moramo dodati vrstico, ki vključuje paketa za delo s slikami:

```
\usepackage{graphics,epsfig}
```

- Slike vključujemo z ukazom `\psfig`, kjer med zavita oklepaja {} zapišemo ime slike in njeno zahtevano širino. Recimo, da želimo vključiti sliko z imenom *UPO*, ki naj bo široka 8 cm. Potem bo koda izgledala takole:

  ```
  \psfig{figure=UPO,width=8cm}
  ```

  V tem primeru mora slika biti v istem imeniku kot datoteka s končnico `tex`, sicer pa moramo ustrezno navesti pot do slike.

- Z ukazom `psfig` lahko vključujemo le slike v formatu PS (PostScript) ali EPS (Encapsulated PostScript). Slike v formatu EPS vidimo tudi v prevedeni LaTeX datoteki na računalniškem zaslonu. Za prikaz slik v formatu PS moramo prevedeno datoteko natisniti na tiskalniku, ki zna interpretirati jezik PostScript, lahko pa jo prevedemo tudi v format PDF in jo nato prikažemo s programom Adobe Reader. Primeri vključevanja slik so v 9. poglavju.

5.3.9 Matematični izrazi

Matematična okolja

Poznamo dva tipa matematičnih okolij:

- vrstičnega in
- sredinjenega.

Vrstično matematično okolje uporabljamo tam, kjer želimo znotraj besedila zapisati nek matematični izraz. V tem primeru matematični izraz vkleščimo med znaka `$`.

Primer kode in njenega prevoda:

```
Komutativnost: $a+b=b+a$
```

Komutativnost: $a + b = b + a$

Sredinjeno matematično okolje pa uporabljamo tam, kjer želimo matematičen izraz izpostaviti, zato se izpiše na sredino, hkrati pa se z vertikalnim razmikom loči od besedila pred izrazom in za njim. Poznamo več sredinjenih matematičnih okolij:

- Osnovno sredinjeno matematično okolje je zelo podobno prej omenjenemu vrstičnemu, le da sedaj matematični izraz vkleščimo med para znakov `$$`. Primer kode in prevoda:

```
$$a+b=b+a$$
```

$$a+b=b+a$$

- Okolje `array` lahko uporabljamo le znotraj matematičnih okolij in je namenjeno ustvarjanju tabel izrazov. Po svoji sintaksi je podobno okolju `tabular`. Razložimo delovanje na naslednjem primeru:

```
$$
\begin{array}{clr}
a+b & a-b & a+b+c\\
1 & 2 & 3
\end{array}
$$
```

V prevodu ta del kode vidimo kot:

$$\begin{array}{clr} a+b & a-b & a+b+c \\ 1 & 2 & 3 \end{array}$$

Komentar kode:

- Osnovno okolje:

```
\begin{array}
\end{array}
```

- Pomen izraza med zavitima oklepajema `{clr}`:
  - znak `c` podaja ukaz sistemu LaTeX, da naj bo izraz v prvem stolpcu sredinjen (*c – center*),
  - znak `l` podaja ukaz sistemu LaTeX, da naj bo izraz v drugem stolpcu poravnan levo (*l – left*),
  - znak `r` podaja ukaz sistemu LaTeX, da naj bo izraz v tretjem stolpcu poravnan desno (*r – right*),
  - celoten izraz pove, da smo ustvarili strukturo s tremi stolpci.
- Ukaz `&` določa mejo med stolpci.
- Ukaz `\\` določa zaključek vrstice.

Okolje največkrat uporabimo pri ustvarjanju matrik, determinant, deljenju izrazov na več vrstic itd. Oglejmo si primer kode in prevoda, nato pa še nekaj ukazov, ki nam pri tem ustvarjanju pridejo prav:

```
$$
\left\{
{\bf X}=
\left[
\begin{array}{clr}
a+b & a-b & a+b+c\\
1 & 2 & 3
\end{array}
\right]+\ldots \right.
$$
```

$$\left\{ \mathbf{X} = \left[\begin{array}{clr} a+b & a-b & a+b+c \\ 1 & 2 & 3 \end{array} \right] + \ldots \right.$$

`\left(`	veliki okrogli oklepaj
`\right)`	veliki okrogli zaklepaj
`\left[`	veliki oglati oklepaj
`\right]`	veliki oglati zaklepaj
`\left\{`	veliki zaviti oklepaj
`\right\}`	veliki zaviti zaklepaj
`\left|`	velika navpična črta na levi strani
`\right|`	velika navpična črta na desni strani
`\left.`	na levi strani ni prikazan oklepaj
`\right.`	na desni strani ni prikazan zaklepaj

Ukaza `\left.` in `\right.` uporabimo za logični zaključek (začetek ali konec) nekega izraza ali njegovega dela, kjer ne želimo imeti para oklepajev.

- LaTeX nam omogoča tudi samodejno številčenje matematičnih izrazov. V ta namen uporabimo okolji `equation` in `eqarray`. Najprej si oglejmo na primeru kode in prevoda okolje `equation`:

```
\begin{equation}
a+b=b+a
\label{Komutativnost}
\end{equation}
```

$$a + b = b + a \tag{5.1}$$

Komentar:

- Okolje deluje podobno kot okolje za sredinjenje `$$`, z razliko, da izrazu predpiše zaporedno številko, ki jo prikaže na desnem robu med okroglimi oklepaji.
- Podobno kot pri okolju `table` in `figure` lahko tudi enačbam damo imena in se nanje nato sklicujemo v besedilu. V našem primeru smo ime `Komutativnost` predpisali z ukazom `\label`:

  ```
  \label{Komutativnost}
  ```

  In kako se na to enačbo sklicujemo v besedilu? Tudi tukaj uporabljamo ukaz `\ref`. Oglejmo si sklic na primeru kode in prevoda:

  ```
  \ldots v ena\v cbi~(\ref{Komutativnost}) je prikazana\ldots
  ```

  ... v enačbi (5.1) je prikazana...

  Tudi tukaj seveda velja, da moramo dokument prevesti dvakrat, da bo ustrezen sklic viden v prevedenem dokumentu. V primeru, da ga prevedemo le enkrat, je v prevedenem dokumentu na mestu vsakega sklica viden vprašaj. In ker sklic predstavlja zgolj ustrezno številko, ga moramo opremiti z ustreznim kontekstom. V našem primeru smo to naredili s parom okroglih oklepajev `()` in besedo `ena\v cbi`.

Sedaj si oglejmo na primeru kode in prevoda še okolje `eqnarray`, ki nam omogoča številčenje vrstic tabel izrazov:

```
\begin{eqnarray}
a+b & = & b+a
\nonumber\\
a+(b+c) & = & (a+b)+c
\label{Asociativnost}\\
a*(b+c) & = & a*b+a*c
\label{Distributivnost}
\end{eqnarray}
```

$$\begin{aligned} a + b &= b + a \\ a + (b + c) &= (a + b) + c && (5.2) \\ a * (b + c) &= a * b + a * c && (5.3) \end{aligned}$$

Komentar:

- Okolje deluje podobno kot okolje `array`, z dvema bistvenima razlikama: vsaki vrstici predpiše zaporedno številko, ki jo prikaže na desnem robu med okroglimi oklepaji in ukazu `\begin{eqnarray}` ne sledi ukaz za oblikovanje (na primer `{clr}`).
- Če določene vrstice ne želimo oštevilčiti, potem pred zaključkom vrstice uporabimo ukaz `\nonumber`.
- Posameznim vrsticam lahko damo imena in se nanje nato sklicujemo v besedilu. Postopek sklicevanja je enak kot pri okolju `array`.

Najpomembnejši ukazi za ustvarjanje matematičnih izrazov

Najprej omenimo znake, ki so pravilno prevedeni le znotraj matematičnega okolja: `| < >`. Če te znake uporabimo v navadnem besedilu, dobimo v prevodu naslednje znake: — ¡ ¿ (izjemi sta sloga pisav `\tt` in `\sc`).

Ukaz za pisanje ulomkov je `\frac`, kjer števec in imenovalec podamo med zavitima oklepajema {}. Oglejmo si primer kode in prevoda:

```
$$
x=\frac{a+b}{a-b}
$$
```

$$x=\frac{a+b}{a-b}$$

In kako ustvarimo potence in indekse? Za potence uporabljamo rezerviran znak `^`, za indekse pa rezerviran znak `_`, ki mu sledi želena oznaka. Če oznaka vsebuje več kot le en znak, jo zapišemo med zavita oklepaja {}. Oglejmo si primer kode in prevoda:

```
$$
x_{22}=a^2+b^2
$$
```

$$x_{22}=a^2+b^2$$

Integrale ustvarimo z ukazom `\int`, odvode pa z ukazom `\prime`.
Oglejmo si primer kode in prevoda:

```
$$
y=\int_0^{33}x^2dx \;\;\;\;\; z=y^{\prime\prime}
$$
```

$$y=\int_0^{33}x^2dx \;\;\;\;\; z=y^{\prime\prime}$$

Ukaz `\;` naredi majhen presledek znotraj matematičnega okolja, saj matematično okolje samodejno izloči vse presledke v izvornem besedilu.

S tem smo osvojili koncept delovanja matematičnega okolja v sistemu LaTeX, zato podajmo ostale priročne ukaze v obliki spiskov:

`\sqrt{x}`	$\sqrt{x}$	`\dot{a}`	$\dot{a}$
`\underline{x}`	$\underline{x}$	`\ddot{a}`	$\ddot{a}$
`\overline{x+y}`	$\overline{x+y}$	`\vec{a}`	$\vec{a}$

`\leftarrow`	$\leftarrow$	`\Leftarrow`	$\Leftarrow$
`\rightarrow`	$\rightarrow$	`\Rightarrow`	$\Rightarrow$
`\leftrightarrow`	$\leftrightarrow$	`\Leftrightarrow`	$\Leftrightarrow$

`\oint`	$\oint$	`\nabla`	∇	`\sum`	$\sum$	`\forall`	$\forall$
`\prod`	$\prod$	`\exists`	$\exists$	`\bigcap`	$\bigcap$	`\neg`	$\neg$
`\bigcup`	$\bigcup$	`\le`	$\le$	`\bigvee`	$\bigvee$	`\ge`	$\ge$
`\bigwedge`	$\bigwedge$	`\subset`	$\subset$	`\pm`	$\pm$	`\subseteq`	$\subseteq$
`\cdot`	$\cdot$	`\equiv`	$\equiv$	`\times`	$\times$	`\propto`	$\propto$
`\Re`	$\Re$	`\in`	$\in$	`\Im`	$\Im$	`\notin`	$\notin$
`\partial`	∂	`\sim`	$\sim$	`\infty`	∞	`\simeq`	$\simeq$
`\emptyset`	$\emptyset$	`\approx`	$\approx$	`\neq`	$\neq$	`\doteq`	$\doteq$

\alpha	α	\iota	ι	\varrho	ϱ
\beta	β	\kappa	κ	\sigma	σ
\gamma	γ	\lambda	λ	\varsigma	ς
\delta	δ	\mu	μ	\tau	τ
\epsilon	ϵ	\nu	ν	\upsilon	υ
\varepsilon	ε	\xi	ξ	\phi	ϕ
\zeta	ζ	\o	o	\varphi	φ
\eta	η	\pi	π	\chi	χ
\theta	θ	\varpi	ϖ	\psi	ψ
\vartheta	ϑ	\rho	ρ	\omega	ω
\Gamma	Γ	\Xi	Ξ	\Phi	Φ
\Delta	Δ	\Pi	Π	\Psi	Ψ
\Theta	Θ	\Sigma	Σ	\Omega	Ω
\Lambda	Λ	\Upsilon	Υ		

Kot ste najbrž opazili, so praktično vsi ukazi zelo razumljivi, saj so to večinoma kar ustrezne angleške besede. Tako zapišemo na primer *podčrtaj* x enostavno kot `\underline{x}`, da dobimo $\underline{x}$.

5.3.10 Za konec še nekaj podrobnosti

Seveda pa to še zdaleč ni vse, kar nam LaTeX ponuja. Tukaj smo omenili le osnove, nič pa nismo spregovorili o ukazih za risanje slik, oblikovanju makrojev ipd. Na samem koncu podajmo le še nekaj koristnih ukazov:

- Z ukazom `\newpage` skočimo na novo stran.
- Z ukazom `\vspace` ukažemo, da LaTeX naredi ustrezen vertikalen presledek med deloma dokumenta. Če želimo na primer narediti vertikalen razmik treh centimetrov, to dosežemo z ukazom `\vspace{3cm}`.
- Z ukazom `\hspace` dosežemo, da LaTeX naredi predpisan horizontalen presledek med deloma dokumenta. Horizontalen razmik za 15 milimetrov dosežemo z ukazom `\hspace{15mm}`.
- Z ukazom `\vfill` dosežemo, da se besedilo, ki temu ukazu sledi, poravna na spodnji rob strani.
- Z ukazom `\hfill` dosežemo, da se besedilo, ki temu ukazu sledi, poravna na desni rob.
- Z ukazom `\mbox` dosežemo, da se besedilo v matematičnem okolju izpiše kot običajno besedilo, saj sicer matematično okolje ignorira vse presledke:

Pravilno: koda → `$z_n=x_n^2+y_n^2 \mbox{ za } n=1..5$`
prevod → $z_n = x_n^2 + y_n^2$ za $n = 1..5$

Napačno: koda → `$z_n=x_n^2+y_n^2 za n=1..5$`
prevod → $z_n = x_n^2 + y_n^2 za n = 1..5$

- Če uporabljamo ukaz `twocolumn` za dvokolonsko oblikovanje besedil in želimo prikazati tabelo ali sliko preko obeh stolpcev (to je možno le na vrhu strani), to naredimo tako, da za ukazom `table` oziroma `figure` zapišemo znak `*`:

```
\begin{figure*}
\ldots
\end{figure*}
```

- LaTeX oziroma TeX ima vgrajena pravila za deljenje besed ameriške angleščine. Ker so pravila za deljenje besed v različnih jezikih različna, LaTeX besed v drugih jezikih ne bo vedno delil pravilno. Sistemu LaTeX je možno dodati pravila za deljenje besed za druge jezike, med drugimi tudi za slovenščino, vendar to presega osnovno znanje. Če beseda na koncu vrstice ni pravilno deljena, lahko sistemu LaTeX pokažemo, kje lahko deli besedo; na primer besedo "delitev" na naslednji način: `de\-li\-tev`.

- LaTeXs pomočjo ustreznih makro paketov omogoča enostaven zapis kemijskih formul, glasbenih notnih zapisov in šahovskih pozicij.

- LaTeX omogoča tudi barvni tisk, kar lahko s pridom izkoristimo, če imamo barvni tiskalnik ali če je izpis namenjen le prikazu na računalniškem zaslonu (npr. za svetovni splet).

- LaTeX je pravzaprav pravi programski jezik, v katerem je možno tudi računati s celimi števili. Kot primer je na sliki 5.34 prikazano besedilo, ki je poravnano vzdolž parametrično podane krivulje. Na sliki 5.35 pa je koledar, ki ga LaTeX pri vsakokratnem prevajanju na novo izračuna za podano leto.

Slika 5.34: Primer nadzora oblike tiskane površine s programom LaTeX

2007

January						
Su	Mo	Tu	We	Th	Fr	Sa
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

February						
Su	Mo	Tu	We	Th	Fr	Sa
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28			

March						
Su	Mo	Tu	We	Th	Fr	Sa
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

April						
Su	Mo	Tu	We	Th	Fr	Sa
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

May						
Su	Mo	Tu	We	Th	Fr	Sa
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

June						
Su	Mo	Tu	We	Th	Fr	Sa
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

July						
Su	Mo	Tu	We	Th	Fr	Sa
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

August						
Su	Mo	Tu	We	Th	Fr	Sa
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

September						
Su	Mo	Tu	We	Th	Fr	Sa
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30						

October						
Su	Mo	Tu	We	Th	Fr	Sa
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

November						
Su	Mo	Tu	We	Th	Fr	Sa
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	

December						
Su	Mo	Tu	We	Th	Fr	Sa
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

Slika 5.35: V sistemu LaTeX lahko izračunamo koledar za poljubno podano leto in oblikujemo njegov izpis

5.3.11 Primerjava vizualnega in logičnega urejanja besedil

Orodja za vizualno urejanje besedil je lažje uporabljati in se jih uporabniki tudi hitreje naučijo. Z vizualnimi orodji je tudi lažje izvajati zahtevno grafično oblikovanje. Primerna so predvsem za kratka besedila, za dolga besedila pa kmalu postanejo preokorna. Nenazadnje kritiki vizualnega urejanja pravijo, da WYSIWYG pravzaprav pomeni "What you see is ONLY what you get" [9]. Skratka, pri bolj zahtevnih besedilih moramo za enotno in predvsem natančno oblikovanje upoštevati tudi njihovo vsebinsko strukturo, kar pa omogoča le logično oblikovanje.

Logično urejanje prav zaradi ločitve vsebine, to je logične strukture besedila, od oblike omogoča konsistentno oblikovanje celotnega besedila na osnovi njegove logične strukture. Logično strukturirana besedila lahko enostavno prevedemo iz ene strukturirane oblike v drugo strukturirano obliko (npr. TeX v HTML ali obratno), ali pa jih ustvarimo z drugimi računalniškimi orodji (npr. ustvarjanje enačb v formatu LaTeX v programu Mathematica). Nenazadnje lahko z logičnim urejanjem besedil dosežemo veliko višjo in konsistentno tipografsko kvaliteto.

Kot primer fleksibilnosti logičnega urejanja besedil si oglejmo naslednji primer ločitve strukture in oblike. V izvornem besedilu v formatu LaTeX bomo pisali notranji produkt na naslednji način: `\np{A}{B}`. Z definicijo makro ukaza `\np` za notranji produkt

```
\newcommand[2]{\np}{(#1,#2)}
```

notranji produkt A in B izpišemo kot (A, B). Le z ustrezno spremembo makro ukaza pa lahko notranji produkt povsod v besedilu izpišemo tudi na druge načine: $(A,\ \ B)$, $(A|B)$ ali $\langle A|B\rangle$.

Prednost logičnega urejanja besedil v primerjavi z vizualnim so tudi lažje prenosljive in veliko manjše datoteke. Izvorne datoteke logično označenih besedil (LaTeX, HTML, XML) so povsem običajne datoteke ASCII, ki ne zastarijo in jih je možno prebrati na praktično katerekoli vrsti računalnika. Datoteke vizualno urejenih besedil pa za razliko poleg samega besedila uporabljajo celo vrsto dodatnih kontrolnih znakov, ki so razumljivi večinoma le programom, s katerimi smo jih uredili. Prenosljivost na druge računalniške platforme je zato veliko težja, težave pa pogosto nastopijo s starejšimi datotekami, ki jih z novimi različicami ne moremo več prebrati. Prisiljeni smo tudi stalno kupovati nove različice vizualnih urejevalnikov, saj s starejšo različico programa običajno ne moremo brati datotek, ki so rezultat novejše različice. Novejše različice povrh vsega za običajnega uporabnika celo ne nudijo nobenih vidnih oziroma funkcionalnih izboljšav. Datoteke vizualno urejenih besedil so nemalokrat 10–100 krat večje kot datoteke istih besedil z logičnimi oznakami.

5.3.12 Naloge

1. Opišite bistvene razlike med vizualnim in logičnim urejanjem besedil! Podajte vsaj dva primera orodij za vsako skupino!
2. Opišite splošen postopek dela s sistemom LaTeX od vpisa kode do prikaza prevoda!
3. Kaj smo povedali o osnovni zgradbi datotek s končnico `tex`?
4. Naštejte vsaj pet okolij sistema LaTeX! Katero okolje smo označili kot okolje vseh okolij?
5. V katerih primerih smo rekli, da moramo za pravilen rezultat LaTeX kodo prevesti dvakrat? Bi si upali sklepati zakaj je temu tako?
6. Kakšna je podobnost in kakšna je razlika med okoljema `verbatim` in `verb`? Opišite še analogijo z okoljema $$ in $!
7. V sistemu LaTeX ustvarite datoteko, katere prikaz bo sledeč:

Rezultati	
Ime in priimek	**Ocena**
Jure Kosec	10
Mateja Cerar	9
Vid Kovač	8

8. V sistemu LaTeX ustvarite datoteko, katere prikaz bo podajal enačbo za izračun povprečne napake ocene globine l v primerjavi z dejansko razdaljo d nad n izbranimi značilkami na sceni:

$$\mathrm{AVG}_{\%} = \frac{\sum_{i=1}^{n} |l_i - d_i|/d_i}{n} \cdot 100\%$$

9. V sistemu LaTeX ustvarite datoteko, katere prikaz bo podajal rotacijsko matriko okoli koordinatne osi x:

$$\mathbf{R}_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix}$$

10. Kakšen bo rezultat prikaza naslednje LaTeX kode po dvakratnem prevodu?

```
\ldots
\section{Sklepi}\label{razdelek:sklepi}
\begin{itemize}
\item \LaTeX\ je zakon!
\item Sklicevanje je sila enostavno:
```

```
smo v razdelku \ref{razdelek:sklepi}.
\item Tudi za na\v{s}tevanje velja enako:
\begin{enumerate}
\item $a+b=b+a$
\item $$$a+b=b+a$
\end{enumerate}
\item \ldots
\item Od danes naprej zato za resno delo uporabljam
le \v{s}e \LaTeX!
\end{itemize}
\ldots
```

11. Nek začetnik se trudi s pripravo dokumenta v sistemu LaTeX.
Zamisli si sledečo tabelo:

\	Mesečni	Letni
Doprinos	15%	26%

in vnese naslednjo kodo:

```
\ldots
/begin{tabular}{|clr|}\hline
\            & Mese\v{c}ni   & Letni /hline
Doprinos     & 15/%       & 26%   /hline
/end{tabular}
\ldots
```

Kje vse se je avtor zmotil? Kako bi izgledala pravilna koda?

5.4 Koristne spletne povezave

1. Spletna stran OpenOffice.org:
`http://www.openoffice.org/`

2. Spletna stran slovenskega OpenOffice.org:
`http://sl.openoffice.org/`
(S te strani je pod licenco GNU/FDL dostopna tudi knjiga [4].)

3. Zavezništvo OpenOffice.org – koristne informacije glede namestitve in uporabe paketa OpenOffice.org:
`http://ooz.agenda.si/`

4. Arhiv CTAN TeX:
`http://www.ctan.org/`

5. Slovenska skupina uporabnikov sistema TeX:
`http://vlado.fmf.uni-lj.si/texceh/texceh.htm`

6. Slovenščina in TEX:
   `http://nl.ijs.si/GNUsl/tex/tslovene/slolang/node98.html`

7. Spletna stran podjetja Adobe Systems Incorporated:
   `http://www.adobe.com/`

8. PostScript:
   `http://www.adobe.com/products/postscript/main.html`

9. Word Processors: Stupid and Inefficient:
   `http://ricardo.ecn.wfu.edu/~cottrell/wp.html`

5.5 Literatura

[1] V. Batagelj and B. Golli. *TEX, Povabilo v TEX, LATEX,* BibTEX, PicTEX. Društvo matematikov, fizikov in astronomov Slovenije, Ljubljana, 1990.

[2] M. Goossens, F. Mittelbach, and A. Samarin. *The LATEX Companion.* Addison-Wesley, Reading, MA, 1994.

[3] M. Goossens and S. Rahtz. *The LATEX Web Companion.* Addison-Wesley, Reading, MA, 1999.

[4] M. Goossens, S. Rahtz, and F. Mittelbach. *The LATEX Graphics Companion.* Addison-Wesley, Reading, MA, 1997.

[5] S. Haugland and F. Jones. *OpenOffice.org 1.0 Resource Kit.* Prentice-Hall PTR, Upper Saddle River, NJ, 2003.

[6] M. Hladnik. *Praktični spisovnik ali šola strokovnega ubesedovanja.* Filozofska fakulteta, Ljubljana, 1991.

[7] D. E. Knuth. *The TEXbook.* Addison-Wesley, Reading, MA, 1994.

[8] D. E. Knuth. *Digital Typography.* CSLI Publications, Stanford, 1999.

[9] L. Lamport. *LATEX A Document Preparation System, User's Guide and Reference Manual.* Addison-Wesley, Reading, MA, second edition, 1994.

[10] G. Leete, E. Finkelstein, and M. Leete. *OpenOffice.org for Dummies.* Wiley Publishing, Inc., Hoboken, NJ, 2003.

[11] R. Ludvik, I. Zajc, and A. Medic. *Hitri vodnik po OpenOffice.org.* Pasadena, Ljubljana, 2003. (Dostopna pod licenco GNU/FDL na: http://openoffice.lugos.si/knjiga/Hitri_vodnik_po_OpenOffice.org_FDL.pdf).

[12] R. C. Parker. *Grafično oblikovanje.* Pasadena, Ljubljana, 1997.

[13] Adobe Systems. *PostScript Language Tutorial and Cookbook*. Addison-Wesley, 1985.

[14] J. Veen. *The Art & Science of Web Design*. New Riders, Indianapolis, IN, 2000.

[15] I. Winkler. *Pisanje strokovnih sestavkov*. Biotehniška fakulteta, Ljubljana, 1998.

[16] K. Možina. *Knjižna tipografija*. Bibliothecaria 13. Univerza v Ljubljani, Filozofska fakulteta, Oddelek za bibliotekarstvo, Ljubljana, 2003.

6

Preglednice

Preglednice so v osnovi elektronska oblika računovodske glavne knjige. Računovodje so že stoletja uporabljali tako imenovane *računovodske glavne knjige*, ki so jih sestavljale strani, oblikovane v vrstice in stolpce, v katere so na pregleden način vnašali različne finančne podatke in transakcije, kot na primer stroške, prihodke, davke itd. Ti podatki v urejeni obliki so jim omogočali analizo in načrtovanje odločitev, predvsem pa medsebojno kontrolo vsot po vrsticah in stolpcih. Seveda pa je bilo to delo zamudno in zaradi ročne obdelave podatkov podvrženo napakam.

Preglednice v elektronski obliki odpravljajo te pomanjkljivosti. V posamezne celice preglednice lahko vnašamo ne le podatke, pač pa tudi formule, s katerimi izračunamo (obdelamo) vnešene podatke. Ob spremembi podatkov v celicah, ki jih obsegajo posamezne formule, se samodejno izračunajo in prikažejo novi rezultati. Formule so lahko relativno preproste, kot je na primer izračun vsote, lahko pa so tudi zelo kompleksne, na primer izračun sistema diferencialnih enačb. Novejše preglednice vsebujejo številne vgrajene funkcije, ki so zaradi lažje preglednosti razvrščene po skupinah: matematične, statistične, tekstovne, finančne, inženirske itd. Poleg tega programska oprema za delo s preglednicami nudi še dodatna orodja, tako da lahko današnje preglednice opišemo kot programsko orodje, ki omogoča celovito urejanje, manipulacijo, analizo in prikaz podatkov. Programska oprema za delo s preglednicami tako običajno vključuje učinkovite in prijazne uporabniške vmesnike, grafično podporo in enostavna orodja za gradnjo in delo s preprostimi oblikami podatkovnih zbirk.

Zgodovinsko gledano predstavlja elektronska oblika preglednice, ki sta jo razvila Dan Bricklin in Bob Frankston ter jo poimenovala *VisiCalc*, prvo resnično uspešno aplikacijo za osebne računalnike. Prvotni program so kljub preprostosti in omejitvam prodali v več kot milijon kopijah. VisiCalc je pripomogel, da so se mnogi ljudje, predvsem poslovneži, odločili za prvi nakup osebnega računalnika.

V 80-ih letih je Mitch Kapor razvil Lotus, ki je hitro postal nov industrijski standard za preglednice. Lotus-1-2-3 je omogočal enostavnejšo uporabo preglednic, pri čemer so mu bila že pridružena orodja za izdelavo grafikonov, diagramov in omogočena je bila izdelava preprostih podatkovnih zbirk. Lotus-1-2-3 se je uveljavil kot orodje za predstavitev podatkov, pa tudi za njihovo kompleksno analizo, kar so omogočale vgrajene matematične funkcije. Lotus je uvedel imenovanje celic, manipulacijo z intervalom celic in makroje. Lotus-1-2-3 še do danes ostaja eden najbolje prodajanih produktov programske opreme vseh časov.

Naslednji mejnik v razvoju preglednic predstavlja Microsoftovo orodje Excel. Prvotno je bil napisan za 512K Apple Macintosh v letih 1984–1985. Excel je bil tudi eden izmed prvih programov za delo s preglednicami, ki je uporabljal grafični vmesnik z meniji in princip direktne manipulacije z miško. Excel je bil eden prvih uporabniških programov, ki so se izvajali pod grafičnim uporabniškim vmesnikom Microsoft Windows.

Razvoj programske opreme za delo s preglednicami gre v smeri izboljšav in izdelave pomožnih programov (angl. *add-in*) za obstoječe programe, kot so Excel, Lotus in drugi, ki omogočajo kompleksne analize (t.i. *goalseeking*), z metodami optimizacije z omejitvami, ki temeljijo npr. na linearnem in nelinearnem programiranju.

6.1 OpenOffice.org Calc

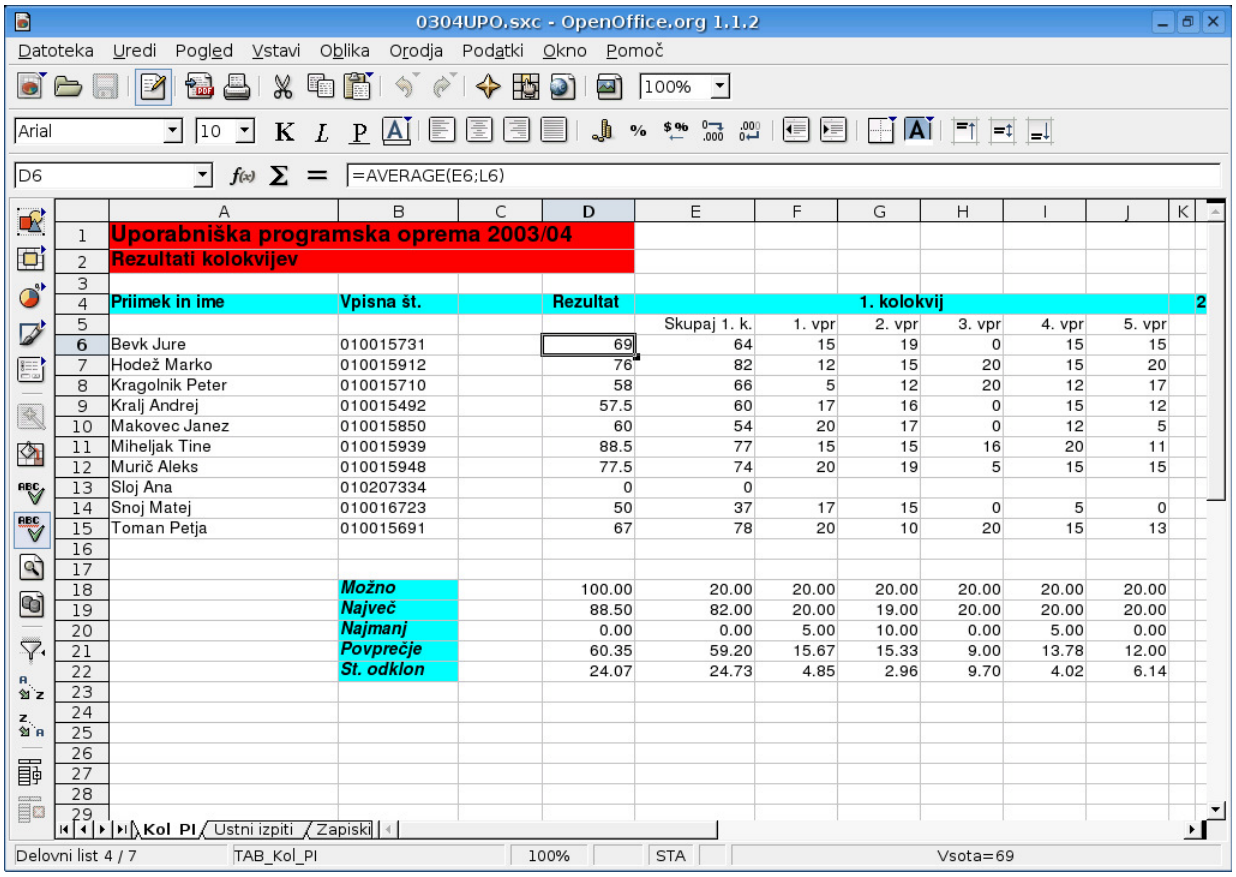
OpenOffice.org Calc je modul paketa OpenOffice.org za delo s preglednicami. V tem poglavju bomo podali napotke za uporabo osnovnih funkcij programa, kot so izdelava preglednic, manipuliranje s podatki in izračunavanje, dotaknili pa se bomo tudi uporabe podatkov kot podatkovne baze in prikazovanja podatkov z grafikoni.

To pa seveda niso vse možnosti, ki jih program ponuja. Z avtomatiziranjem izračunov in programiranjem vnosnih pogovornih oken z vgrajenim makro jezikom OpenOffice.org Basic lahko hitro izdelamo preproste aplikacije.

6.1.1 Prvo srečanje s preglednico

Novo preglednico lahko odpremo na več načinov: v terminalskem oknu z ukazom `oocalc`, preko namizja in menijev operacijskega sistema, ali pa preko menijske izbire poljubnega že odprtega modula OpenOffice.org, kjer izberemo "Datoteka→Nova→Spreadsheet". Primer preglednice v oknu OpenOffice.org Calc je na sliki 6.1.

Preglednico sestavlja več posameznih delovnih listov (angl. *worksheet*), vsak delovni list pa vsebuje celice, organizirane v vrstice in stolpce. Vsaka

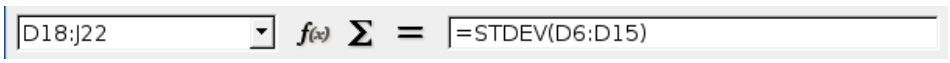


Slika 6.1: Delo s preglednico v programu OpenOffice.org Calc

celica v delovnem listu je enolično določena s kombinacijo oznake stolpca in številke vrstice, npr. `D6`.

Oznake stolpcev in vrstic so vidne ob levem in zgornjem robu dokumenta. Če kliknemo na oznako, izberemo celo vrstico ali stolpec. Zavihke na dnu dokumenta uporabljamo za preklapljanje med delovnimi listi.

Vrstice so označene s številkami od `1` do `32000`. Stolpci se vrstijo od `A` preko `AA` do `IV`.



Slika 6.2: Vrstica za vnos

V tabeli 6.1 vidimo vsebino glavne orodne vrstice, ki se nahaja levo od oznak vrstic na sliki 6.1. Pod predmetno vrstice se nahaja vnosna vrstica (slika 6.2), ki vsebuje dve polji: v prvem vidimo referenco na trenutno izbrano območje, v drugo polje pa vnašamo podatke in formule. Vnašamo jih lahko tudi neposredno v trenutno izbrano celico; vnos bo viden tako v

Ikona	*Ime orodja*	*Opis*
	“Vstavi”	Vstavljanje objektov (okvirjev, slik itd.)
	“Vstavi celice”	Vstavljanje celic v dokument
	“Vstavi predmet”	Vstavljanje predmetov iz drugih aplikacij
	“Funkcije risanja”	Gumbi za risanje likov
	“Funkcije obrazca”	Funkcije za delo z obrazci
	“Samooblikovanje”	Samodejno oblikovanje izbora
	“Izberi teme”	Izbor videza preglednice
	“Črkovalnik”	Preverjanje črkovanja
	“Samodejno prev. črkovanja”	Vklop/izklop funkcije preverjanja
	“Iskanje”	Iskanje in zamenjava
	“Viri podatkov”	Delo z zbirkami podatkov
	“Filtriranje”	Samodejni filter
	“Razvrsti A→Z”	Naraščajoče razvrščanje
	“Razvrsti Z→A”	Padajoče razvrščanje
	“Vstavi skupino”	Združevanje
	“Razstavi skupino”	Razdruževanje

Tabela 6.1: Orodja v glavni orodni vrstici programa OpenOffice.org Calc

okviru celice kot v vnosnem polju. Z vnosnim poljem si lahko pomagamo pri vnosu daljših podatkov, tako da nam celice ni potrebno povečati. Vnosna vrstica ima poleg teh polj še nekaj koristnih gumbov: gumb “$f(x)$” omogoča vodeno izdelavo kompleksnejših formul, gumb “=” preklopi vrstico v urejevalni način, “$\sum$” pa skonstruira formulo za vsoto vsebine izbranih celic.

Po preglednici se sprehajamo s smernimi tipkami ali miško. Strani preskakujemo s tipkama `<Page Up>` in `<Page Down>`, podobno kot pri urejevalnikih besedil.

Delo z obsežnimi preglednicami si lahko olajšamo tako, da delovni list razdelimo na več delov, po katerih individualno navigiramo. Vsak delovni

list lahko razpolovimo z uporabo horizontalnega in vertikalnega razdelilnika; horizontalni razdelilnik se nahaja desno od horizontalnega drsnika, vertikalni razdelilnik pa nad vertikalnim drsnikom. Dovolj je, da na razdelilnik kliknemo in ga povlečemo do želene širine. Delovni list tako razdelimo na največ štiri površine. Razdelitev velja samo za trenutni delovni list. Če nam to še vedno ni dovolj ali pa če bi radi delali z več delovnimi listi naenkrat, si pomagamo z izbiro "Okno→Novo okno".

Ko odpremo nov dokument, vsebuje tri delovne liste. V enem dokumentu imamo lahko do 256 delovnih listov. Dodajamo jih z ukazom "Vstavi→Delovni list" ali pa kar s pomočjo priročnega menija, ki ga prikličemo z desnim klikom na jeziček lista. Priročni meni nam omogoča še vrsto drugih operacij nad delovnimi listi, npr. preimenovanje lista.

Kot smo že omenili, vrstice in stolpce izbiramo s klikom na oznake ob robu okna, npr. `V` ali `2`. Celo preglednico izberemo s klikom na polje v zgornjem levem kotu, med oznakami za stolpce in tistimi za vrstice.

Polja lahko izbiramo na več načinov. Najlažji je z uporabo akcije vleci in spusti: kliknemo na začetni kot pravokotnega območja, ki ga želimo izbrati, gumb držimo in izbrano območje razširimo do poljubne velikosti. Pri označevanju čez več strani si pomagamo s tipko `<Shift>`: kliknemo prvo polje in hkrati držimo tipko `<Shift>`, nato pa kliknemo na zadnje polje. Več ločenih pravokotnih območij združimo v en izbor s tipko `<Control>`. Zahtevnejše izbore pa lahko vpisujemo kar v polje v vnosni vrstici, npr. `V2:Z201`.

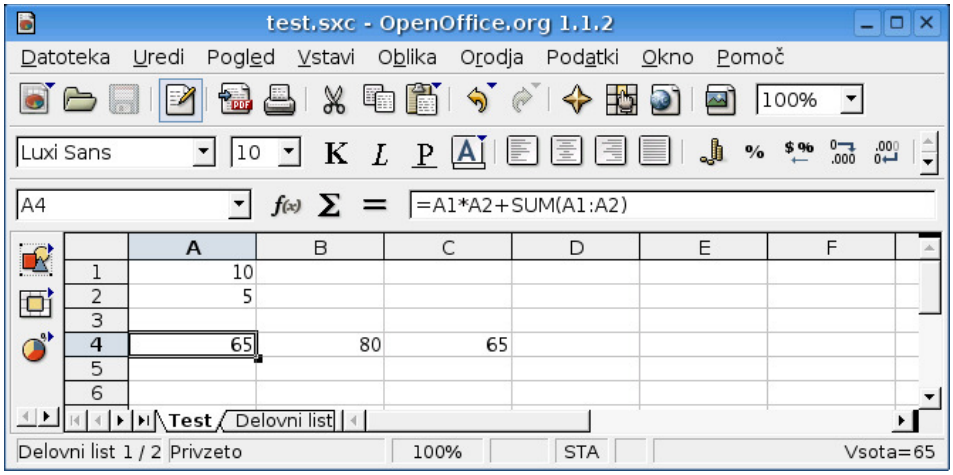
Celica je lahko v pregledovalnem ali urejevalnem načinu. V pregledovalnem načinu se s smernimi tipkami premikamo med celicami, pri čemer ne vidimo v celoti dolgih vsebin znotraj celic. Zato lahko celico postavimo v urejevalni način (z dvojnim klikom na celico ali s tipko `<F2>`), kjer vidimo celotno vsebino celice in jo lahko urejamo kot v navadnem urejevalniku. Pritisk na tipko `<Enter>` bo vsebino vnesel in premaknil izbor na naslednjo celico.

Z bloki celic delamo podobno kot z bloki besedila v programu OpenOffice.org Writer. Uporabljamo torej akcijo vleci in spusti ter kombinacije tipk `<Control>+<C>` za kopiranje, `<Control>+<V>` za prilepljanje, `<Delete>` za brisanje, ali pa ustrezne podizbire v meniju "Uredi". Vsako akcijo lahko prekličemo s `<Control>+<Z>` ali "Uredi→Razveljavi".

6.1.2 Celice in njihova vsebina

V celice lahko vnašamo podatke ali formule. Formule ob izračunu izpišejo podatke, ki zapolnijo celico. Vsebino v obliki formule vidimo le v vnosni vrstici ali ko vstopimo v urejevalni način. Da bi celice, ki vsebujejo formule, izpostavili, vklopimo opcijo "Pogled→Poudarjanje vrednosti", ki

vsebino teh celic obarva zeleno.



Slika 6.3: Izračuni s formulami

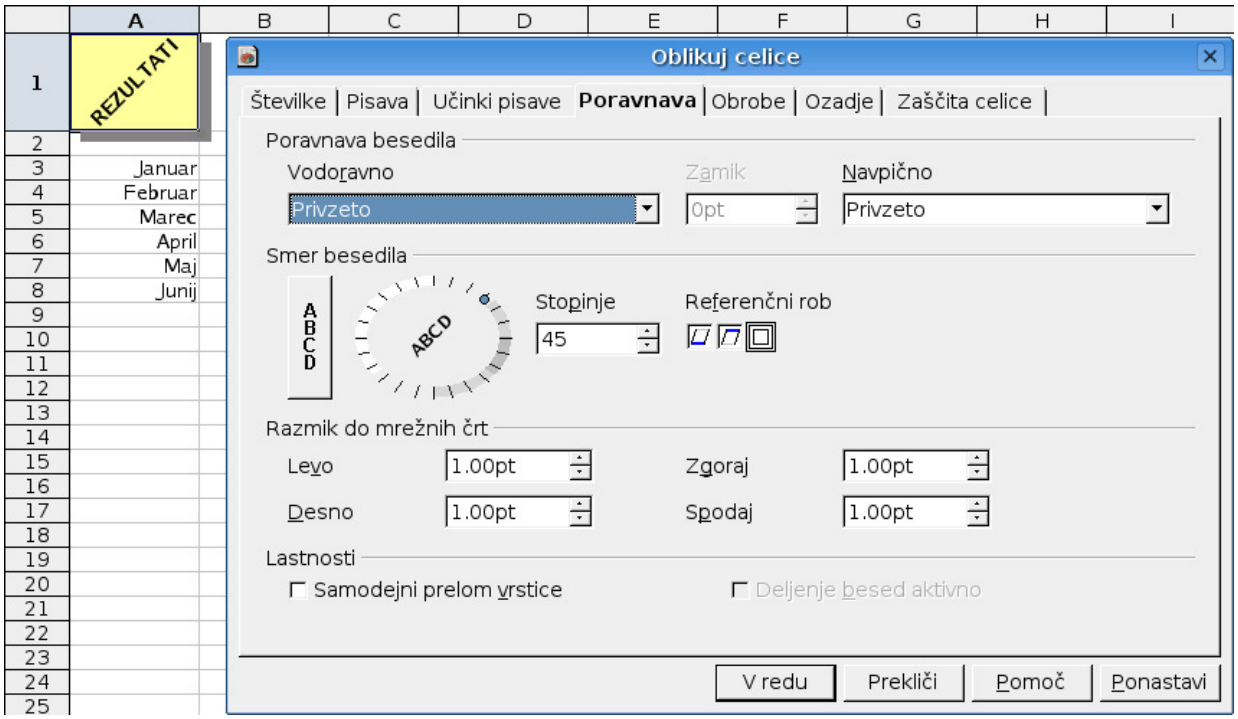
Formulo vnesemo v celico tako, da vnos začnemo z znakom `=`. Primer izračuna lahko vidimo na sliki 6.3. V celico `A1` smo vpisali vrednost 10, v celico `A2` pa vrednost 5. Zmnožek celic izračunamo tako, da v celico `A4` vtipkamo `=A1*A2` ter pritisnemo `<Enter>`. Rezultat 50 se pokaže v celici `A4`. V formulah lahko poleg aritmetičnih izrazov uporabimo tudi funkcije: `=A1*A2+SUM(A1:A2)` bo prejšnjemu rezultatu prištel vsoto vrednosti v `A1` in `A2`, rezultat bo torej 65. V celico `B4` vpišimo `=SUM(A1;A2;A4)`; rezultat bo vsota teh treh celic. V `C4` vpišimo `=B4-(A1+A2)`. Rezultat bo ponovno 65. Poskusimo, kaj se zgodi, ko spremenimo vsebino celice `A1`. Vidimo, da se vsi izračuni osvežijo z novimi vrednostmi.

Vsebina preglednice seveda ni omejena na števila. V nadaljevanju bomo našteli in opisali tipe podatkov, ki jih OpenOffice.org Calc podpira. Prikazovanje podatkov v celici lahko poljubno nastavimo: besedilo oblikujemo, numerične podatke izpisujemo v različnih oblikah, datume in denarne zneske pa prilagodimo lokalnim zahtevam. Nastavitve spreminjamo preko pogovornega okna, ki ga prikličemo bodisi z izbiro “Oblika→Celice” ali pa iz priročnega menija posamezne celice (izbira “Oblikuj celice”, slika 6.4).

V oknu lahko nastavljamo vizualni izgled (“Pisava”, “Učinki pisave”, “Poravnava”, “Obrobe”, “Ozadje”), varnostne parametre (“Zaščita celice”) ter obliko izpisa (“Številke”).

Besedilo

OpenOffice.org Calc definira tip vsebine celice šele tedaj, ko končamo z vnosom. Če vsebina ni podobna številu, datumu, uri ali formuli, jo sprejme kot besedilo. Primeri besedila so na primer “`Leto 2000`” ali



Slika 6.4: Pogovorno okno za nastavitev celice

"`Tabela 1`", pa tudi "`Vsebina te celice je zelo dolga poved; tudi dolge povedi lahko vtipkamo v celico preglednice.`".

Če želimo enega od tipov podatkov vnesti kot besedilo, za prvi znak vtipkamo enojni narekovaj, npr. '2000.

Nad besedilom v celici lahko uporabimo kar nekaj operacij, znanih iz urejevalnikov besedil. Besedilo lahko oblikujemo (trdo oblikovanje), prelamljamo vrstice, poravnavamo itd. Pri tem uporabljamo "Pisava", "Učinki pisave", "Poravnava", "Obrobe" ter "Ozadje" v pogovornem oknu *Oblikuj celice* (slika 6.4).

Besedilo lahko uporabimo tudi v formulah, saj OpenOffice.org Calc podpira vrsto funkcij za delo z besedilom. Besedilo v celicah `A2` in `B3` lahko združimo v celici `C4` tako, da vanjo vtipkamo `=CONCATENATE(A2;B3)`. Podrobnejši opis funkcij za delo z besedilom najdemo v nadaljevanju.

Števila

Števila lahko vnašamo v več oblikah. Lahko jih vnesemo v običajnem zapisu, kot cela ali decimalna stevila, npr. 53, +5.331, 0, −2000.

Odstotke lahko pišemo kar s pomočjo znaka za odstotek, npr. 25%.

Vnašamo lahko tudi ulomke. Vnos 1 3/7 se bo v celico zapisal kot vrednost $1\frac{3}{7}$.

Uporabimo lahko tudi znanstveno notacijo s plavajočo vejico: $10e24$ ali $-53.323e9$.

Pomembno je razlikovati med obliko izpisa števila in vnešeno vrednostjo, saj se natančnost kljub omejitvi izpisa ne spreminja. Ko v celico vpišemo število 2.234534, na zaslonu pa vidimo le 2.23, moramo torej vedeti, da bo program dejansko operiral s polnim številom decimalk.

Vrednosti, ki imajo osnovo v denarju, zapišemo z uporabo oznake valute, npr. $20000. Izpis v celici bo `$20,000.00`, saj se valuti doda vejica ob tisočicah in dve decimalki. Vrsto valute lahko seveda spreminjamo, prav tako pa tudi način izpisa.

Obliko izpisanega števila nastavljamo v pogovornem oknu *Oblikuj celice*, zavihek "Številke". V polju "Kategorija" izberemo vrsto podatka – število, odstotek, valuta, datum, čas, število v znanstveni notaciji, ulomek, logična vrednost ali besedilo. V polju "Oblika" nato izbiramo med vnaprej definiranimi oblikami. Videz vsebine si lahko ogledamo desno od polja "Oblika".

Obliko zapisa lahko tudi sami spreminjamo, za to pa moramo vedeti, kako so zapisani. Vsaka oblika je sestavljen iz posebnih znakov:

- `#`: predstavlja števko; če ima število več števk kot je znakov `#`, bo izpis zaokrožen.
- `0`: števka; če ima število manj števk, se na tistem mestu izpiše ničla.
- `$`: znak za dolar.
- `.`: pika pri tisočicah ali pika pri datumu.
- `,`: decimalna vejica
- `%`: znak za odstotek.
- `?/?`: ulomek.
- `E e`: znak za eksponent.
- `/ - :` : ločila pri datumih in uri.
- `[barva]`: barva znakov.
- `D M N Y Q W`: oblikovanje datumov: `D` – dan, `M` – mesec, `N` – ime dneva, `Y` – leto, `Q` – četrtletje, `W` – teden.
- `H M S`: oblikovanje ure: `H` – ura, `M` – minuta, `S` – sekunda.
- `CCC`: oznaka valute.

Novo obliko vnesemo v polje "Koda oblike". Ko smo z njim zadovoljni, vnos potrdimo in oblika se bo shranil pod trenutno vrsto podatka ("Kategorija"), hkrati pa še v "Uporabniško določeno".

`1784,321` se bo z obliko `#.###,#` izpisala kot `1,784.3`, z obliko `0####,#000` pa kot `01784,3210`. Poglejmo, kako bi definirali obliko, ki bo pozitivna števila izpisoval zeleno, dodal piko pri tisočicah, dodajal vodilne ničle do drugega mesta (desetic) in zmeraj izpisal tri decimalna mesta, pri negativnih številih pa prikazal samo celi del z vodilnimi ničlami do desetic, dodal piko za tisočice in izpis obarval rdeče:

```
[GREEN]#.#00,000;[RED]-#.#00
```

Obliko s plavajočo vejico lahko definiramo kot `0.00E+00`, izpis v ulomku z dvomestnim imenovalcem pa definiramo kot `?/??`.

Pod numerične podatke sodijo tudi denarni zneski, ki so praviloma vedno izpisani v določeni valuti. Ker imajo različne države različne konvencije o zapisovanju denarnih zneskov, moramo nastavitve prilagoditi svojim potrebam. V pogovornem oknu *Oblikuj celice* nastavimo znotraj zavihka "Številke" opcijo "Jezik", ki vpliva na postavitev ločil, izpis datuma, časa in valute. Priporočljivo je, da valuto nastavimo sami, saj se sicer lahko zgodi, da se bo pri prenosu preglednice na drug računalnik z drugačnimi sistemskimi nastavitvami spremenila oznaka valute, znesek pa bo isti. V obliki valute oznako izbrane valute dobimo s pomočjo niza `CCC`.

Datumi

Datum vpišemo bodisi kot `8.14.2004`, `8/14/2004` ali pa `8-14-2004`. Dvoštevilčnemu zapisu leta se zaradi problema Y2K raje odpovejmo. Tudi datume lahko priredimo lokalnim zahtevam. Domač način izpisa bi dosegli z obliko `D.M.YYYY`. Z datumi lahko tudi računamo: denimo, da v celico `A1` vpišemo nek datum. Koliko ur je od tedaj poteklo, izvemo z `=(NOW()-A1)*24`.

Čas

Čas zapišemo kot `12:33:59`. Vnos `43242:143:32` OpenOffice.org Calc preračuna v ustrezno število ur, minut in sekund. Obliko definiramo z uporabo že omenjenih oznak `H`, `M`, in `S`, kot npr. `HH:MM:SS`.

Poleg naštetih vrst lahko celica vsebuje še logično vrednost – `TRUE` ali `FALSE`.

6.1.3 Naslavljanje celic

Omenili smo že, da je celica v delovnem listu enolično določena z oznako stolpca in vrstice. Vendar se lahko nanjo sklicujemo tudi iz drugih delovnih listov ali celo iz drugih preglednic.

Na celico v drugem delovnem listu se sklicujemo tako, da navedemo ime lista in piko, npr. `=ImeLista.C4`. Če ime lista vsebuje tudi presledke, moramo ime zapisati med enojne narekovaje, npr. `='Ime lista'.C4`. Za celico v drugi preglednici moramo navesti še ime datoteke, npr. `='test.sxc'#$ImeLista.C4`. Ime datoteke mora biti navedeno med enojnimi narekovaji, pred imenom lista pa vpišemo znaka `#$`.

Celice lahko tudi poimenujemo s pomočjo izbire "Vstavi→Imena→Določi". Vnesemo ime, npr. `Rezultat` in kliknemo "V redu". Ime lahko sedaj uporabljamo namesto naslova, npr. `Rezultat` namesto `B10`.

Poimenujemo lahko tudi območja; najprej območje izberemo, nato ga poimenujemo. Območja in celice z imeni lahko izbiramo iz padajočega menija v vnosni vrstici.

Absolutno in relativno naslavljanje

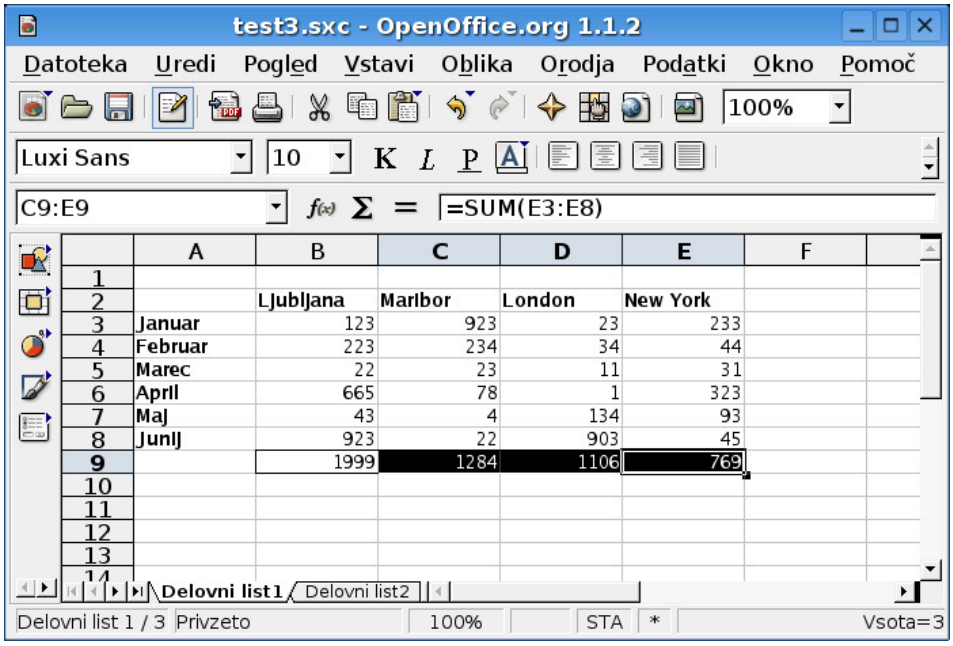
Na celice se sklicujemo v formulah. Če želimo v drugo celico samo prepisati vsebino neke celice, lahko to storimo tako, da vpišemo npr. v celico `A4` formulo `=A1`, kjer je `A1` naslov prve celice.

Denimo, da hočemo isto ponoviti še v celici `B5`. Označimo torej celico `A4` in vsebino prenesemo v celico `B5`. Vendar vsebina celice `B5` ne bo enaka vsebini celice `A4`. Ko pogledamo formulo v `B5`, vidimo, da se je njena vsebina spremenila ustrezno premiku, torej na `=B2`.

Tak način naslavljanja imenujemo **relativno naslavljanje**. Ko vnašamo formule v taki obliki, podajamo relativne naslove celic, ki so odvisni od tega, v katero celico vpisujemo. Na sliki 6.5 vidimo primer izračuna, ko to lastnost izkoristimo za prenos formule v drug del delovnega lista, kjer opravlja isto funkcijo na drugih podatkih. V celico `B9` vpišemo formulo `=SUM(B3:B8)`, jo izberemo in skopiramo v celice `C9` do `E9`, kjer dobimo nove formule, na primer `=SUM(E3:E8)` v celici `E9`.

Če želimo naslov celice zapisati absolutno, moramo uporabiti **absolutno naslavljanje**. Del naslova, ki ga deklariramo kot absolutnega, opremimo z znakom `$`. Za popoln absolutni naslov postavimo `$` tako pred oznako vrstice kot pred oznako stolpca, na primer `$A$4`.

Prejšnjo tabelo lahko spremenimo tako, da v celico `B9` vpišemo formulo `=SUM($B3:$B8)`, jo izberemo in skopiramo v celice `C9` do `E9`, kjer dobimo nove formule, npr. v `E9` `=SUM($B3:$B8)`. Vidimo, da se naslovi celic v



	A	B	C	D	E	F
1						
2		Ljubljana	Maribor	London	New York	
3	Januar	123	923	23	233	
4	Februar	223	234	34	44	
5	Marec	22	23	11	31	
6	April	665	78	1	323	
7	Maj	43	4	134	93	
8	Junij	923	22	903	45	
9		1999	1284	1106	769	
10						
11						
12						
13						
14						

Slika 6.5: Uporaba relativnega naslavljanja

formuli niso spremenili. Da nam bo princip delovanja še jasnejši, skopirajmo celico `B9` v celico `C10`. Nova formula bo `=SUM($B4:$B9)`. V formuli so se spremenili le naslovi vrstic, saj niso zapisani z absolutnim naslavljanjem.

Absolutno lahko naslavljamo tudi z imeni celic ali območij. Denimo, da smo v prejšnjem primeru območje `B3:B8` poimenovali z `Ljubljana`. Formulo z absolutnim naslovom v `B9` sedaj zapišemo kot `=SUM(Ljubljana)`.

6.1.4 Še o formulah in funkcijah

Poznavanje možnosti, ki jih ponujajo formule, je bistveno za učinkovito rabo preglednic. Če vemo, kako formulirati kompleksne izračune, so možnosti tako rekoč neomejene.

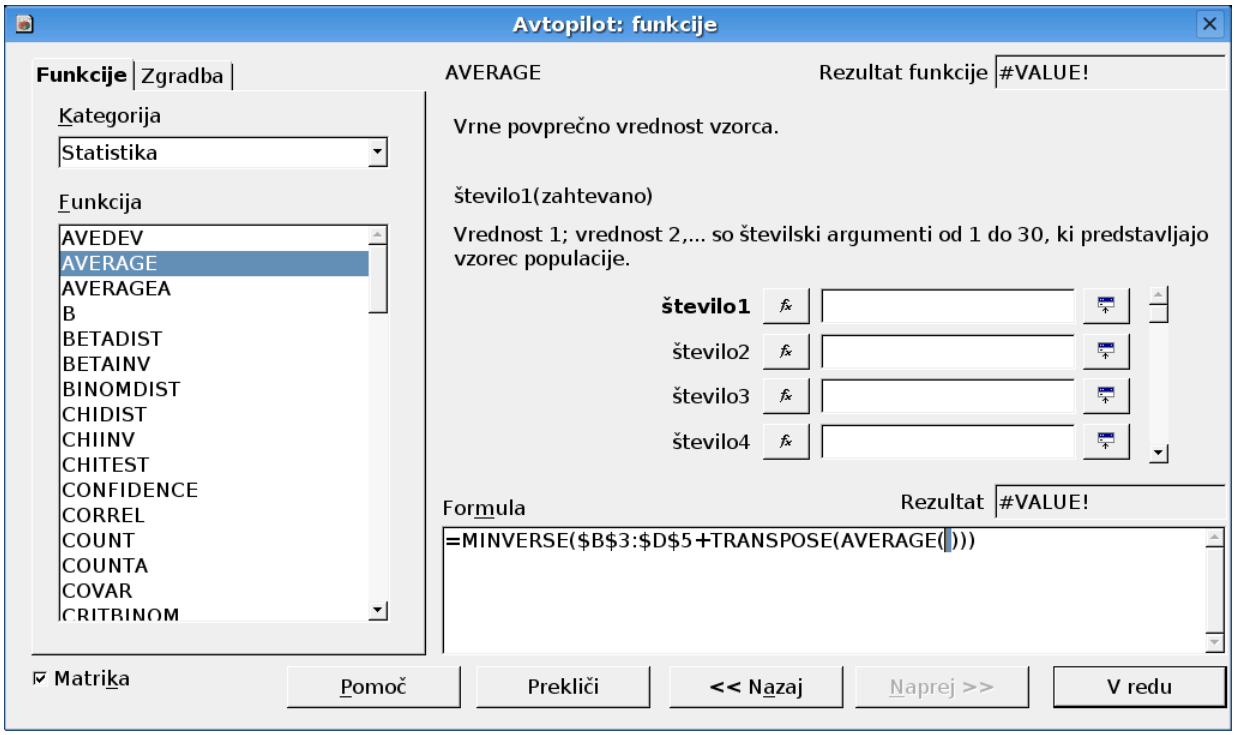
V nadaljevanju bomo opisali najosnovnejše principe formul in navedli nekaj pomembnejših funkcij. Vse funkcije so podrobneje obrazložene v pomoči ("Pomoč→Vsebina").

Opisali smo že, kako formulo vnesemo v celico. Poglejmo podrobneje, kako so formule sestavljene. V tabeli 6.2 so zbrani dovoljeni matematični in relacijski operatorji.

Najvišjo prioriteto ima potenciranje, nato množenje in deljenje, nazadnje pa seštevanje in množenje, vse od leve proti desni.

Aritmetični		*Relacijski*	
+	seštevanje	=	enakost
-	odštevanje	<	manjše
*	množenje	>	večje
/	deljenje	<=	manjše ali enako
^	potenciranje	>=	večje ali enako
		<>	različno

Tabela 6.2: Matematični operatorji v orodju OpenOffice.org Calc



Slika 6.6: Čarovnik za sestavljanje formul

Pomagamo si lahko tudi z oklepaji, npr. `=A1*(11-C3)^2`. Celice v formuli naslavljamo tako, kot smo opisali v prejšnjem poglavju. Funkcije podajamo z imenom in argumenti, ki jih navedemo v oklepaju, na primer

```
=AVERAGE(10.1;$A$3;B$2:B$10;Ljubljana)
```

Argumenti so ločeni s podpičjem. Argumenti so lahko števila, besedilo, datum, čas, logične vrednosti ter funkcije, naslovi celic ali območij, ki vsebujejo ali vračajo te tipe podatkov. Če funkcija nima argumentov, jo podamo s praznimi oklepaji, npr. `PI()`.

Funkcije lahko tudi gnezdimo:

```
=SQRT(SUM($A$1:$A$10;SQRT($B$2))*COUNT($A$1:$A$10)+1)
```

Pri sestavljanju funkcij si lahko pomagamo z že omenjenim čarovnikom (izbira gumba $f(x)$ v vnosni vrstici ali "Vstavi→Funkcija"), ki ga vidimo na sliki 6.6.

Formulo vpisujemo v polje "Formula". Pri vnosu funkcij nam čarovnik pomaga tako, da opiše izbrano funkcijo ter poda polja za argumente. V padajočem meniju "Kategorija" izberemo tip funkcije ("Matematika", "Logika", "Besedilo" ipd.), kar nam omeji možne izbire v polju "Funkcija". Ko funkcijo izberemo, lahko argumente vpisujemo v temu namenjena polja. Hierarhično zgradbo formule si ogledamo v zavihku "Zgradba".

Izbiramo lahko med 375 funkcijami, ki so razdeljene v enajst skupin. Poglejmo si nekaj najpogosteje uporabljenih funkcij:

Matematične funkcije

`ABS(N)` Vrne absolutno vrednost števila, podanega z `N`.

`EXP(N)` Vrne e^N, torej vrednost eksponentne funkcije za vrednost, podano z `N`.

`LN(N)` `LOG10(N)` `LOG(N;B)` Naravni in desetiški logaritem ter logaritmi s poljubno osnovo. Funkciji `LOG` podamo kot prvi argument število, kot drugi pa osnovo.

`POWER(B;P)` Vrne B^P.

`SQRT(N)` Kvadratni koren.

`PRODUCT(N1;N2;...N30)` `SUM(N1;N2;...N30)` Produkt in vsota argumentov `N1`...`N30`.

`RAND()` Generira naključno število med 0 in 1.

`SIN(N)` `COS(N)` `TAN(N)` Trigonometrične funkcije.

`PI()` Število π.

Statistične funkcije

`AVERAGE(N1;N2;...N30)` Vrne srednjo vrednost vzorca, podanega z `N1...N30`.

`MEDIAN(N1;N2;...N30)` Mediana vzorca, podanega z `N1...N30`.

`COUNT(N1;N2;...N30)` Vrne število numeričnih vrednosti, določenih z argumenti.

`COUNTA(N1;N2;...N30)` Vrne število kakršnihkoli vsebin, podanih z argumenti.

`MAX(N1;N2;...N30)` Vrne največjo vrednost med števili, ki jih določajo argumenti.

`MIN(N1;N2;...N30)` Vrne najmanjšo vrednost med števili, ki jih določajo argumenti.

`LARGE(N1;N2;...N30;K)` Vrne `K`-to največje število iz vzorca `N1...N30`.

`SMALL(N1;N2;...N30;K)` Vrne `K`-to najmanjše število iz vzorca `N1...N30`.

Funkcije, ki delujejo nad besedilom

Denimo, da v celico `A1` vpišemo `Fata`, v celico `B1` pa `Morgana`.

`CONCATENATE(T1;T2;...T30)` Združi besedila, podana s `T1...T30`.
Primer: `=CONCATENATE(A1; B1)` vrne rezultat `FataMorgana`.

`LOWER(T)` `UPPER(T)` Vse črke spremeni v male/velike črke. Primer:
`=LOWER(CONCATENATE(A1; B1))` vrne `fatamorgana`;
`=UPPER("fatamorgana")` vrne `FATAMORGANA`.

`PROPER(T)` Vsako besedo zapiše z veliko začetnico. Primer:
`=PROPER(LOWER(CONCATENATE(A1; B1)))` vrne `Fatamorgana`.

`TEXT(N;F)` Podano število `N` pretvori v besedilo in sicer v obliki, določeni z `F`. Primer:
`=TEXT(-99000.1;"#,#00.000")` vrne besedilo `-99,000.100`.
Obliko moramo podati kot tekst, torej med navednicami.

Datumske funkcije

Na sistemih Linux sta čas in datum shranjena kot t.i. zaporedno število (angl. *serial number*), ki določa število sekund od nekega trenutka dalje (navadno od 30.12.1899 dalje). Vsa preračunavanja z datumi uporabljajo zaporedna števila, zato moramo znati pretvarjati med enim in drugim zapisom.

`DATE(Y;M;D)` Pretvori dani datum v zaporedno število. Primer: `=DATE(2001;9;1)` vrne zaporedno število `37135`. Število v celici ni vidno, zato moramo celici določiti številsko obliko.

`TODAY()` Vrne zaporedno število trenutnega datuma.

`DAY(N)` Vrne dan v mesecu za podano zaporedno število.

`MONTH(N)` Vrne mesec za podano zaporedno število.

`YEAR(N)` Vrne leto za podano zaporedno število.

`DAYS(N1;N2)` Vrne število dni med dvema datumoma, določenima z zaporednima številoma. Primer: `=DAYS(DATE(2001;1;1);DATE(2002;1;1))` vrne `-365`.

Logične funkcije

`TRUE()` Vrne logično vrednost `TRUE`.

`FALSE()` Vrne logično vrednost `FALSE`.

`AND(C1;C2;...C30)` Konjunkcija: `TRUE`, če so izpolnjeni vsi pogoji `C1...C30`.

`OR(C1;C2;...C30)` Disjunkcija: `TRUE`, če je izpolnjen vsaj en pogoj.

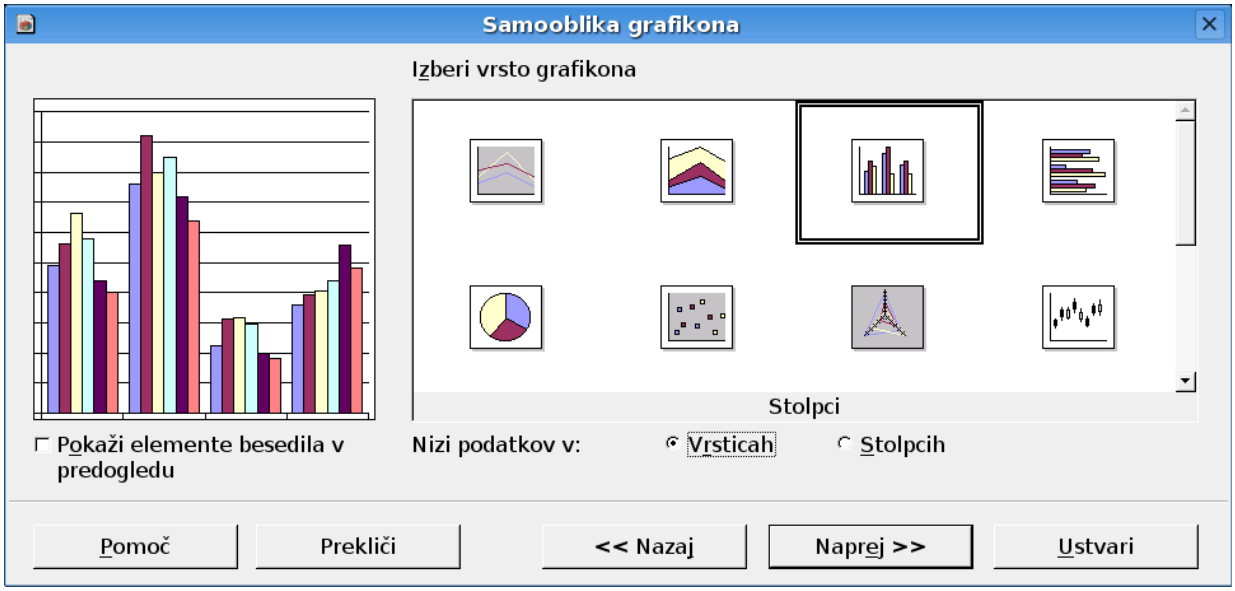
`NOT(C1)` Logična negacija: `TRUE`, če je `C1 FALSE`.

`IF(C;T;F)` Vrne vrednost `T`, če je pogoj `C` izpolnjen, sicer vrne vrednost `F`. Primer:
`=IF(C9>10; C9; IF(C9>=0; ”Pozor! Stanje<10!”; ”Pozor! Negativno stanje!”))`.

6.1.5 Izdelava grafikonov

Ob številnih podatkih, organiziranih v tabele, se pojavi potreba po vizualizaciji [5, 6, 7], ki poveča njihovo preglednost in tako olajša analize in odločitve. Grafikone delimo na dvo in tridimenzionalne. Predstavljanje multidimenzionalnih in dinamičnih podatkov pa še vedno predstavlja trd oreh, s katerim se ukvarja veliko število raziskovalcev.

Podatke, ki jih bomo vključili v grafikon, je potrebno najprej izbrati. Če ima tabela tudi oznake (stolpcev, vrstic), izberemo tudi te. Pogoj je le, da se celice oznak in podatkov stikajo.

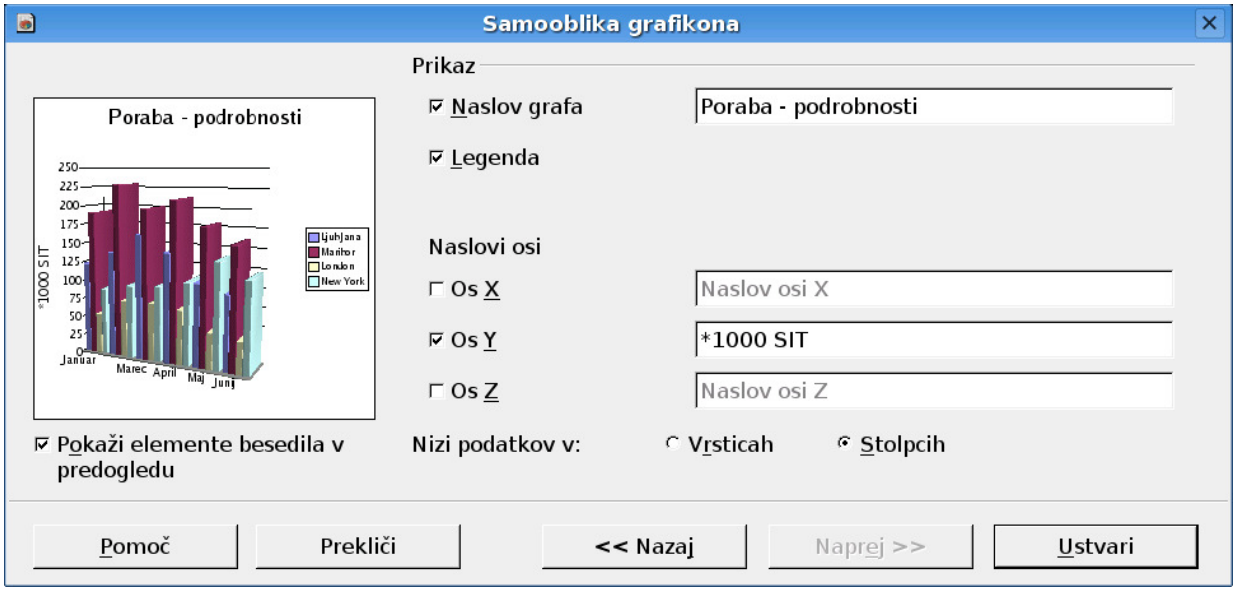


Slika 6.7: Izbira vrste grafikona

Ko podatke označimo, izberemo “Vstavi→Grafikon”. Prikaže se prvo pogovorno okno čarovnika, v katerem lahko popravimo začetne nastavitve bloka in oznak. V naslednjem pogovornem oknu (slika 6.7) izberemo vrsto grafikona. Na voljo imamo dvodimenzionalne in tridimenzionalne vrste grafikonov. V “Nizi podatkov v” določimo, kako naj OpenOffice.org Calc prikaže podatke: opcija “Vrsticah” združi podatke po vrsticah naše tabele, opcija “Stolpcih” pa po stolpcih naše tabele. V tretjem pogovornem oknu pod “Mrežne črte” izberemo, na katerih oseh želimo pravokotno mrežo. Tukaj lahko izberemo tudi različico izbrane vrste grafikona. Četrto okno pa nam omogoča dodajanje in spreminjanje label (slika 6.8).

Ko grafikon postavimo v preglednico, ga lahko vedno izberemo ter spreminjamo njegove lastnosti. Izberemo lahko tudi posamezne elemente grafikona, kot so osi, labele, stolpci in jih poljubno spreminjamo.

Na sliki 6.9 vidimo primer preglednice z dvema grafikonoma. Levi



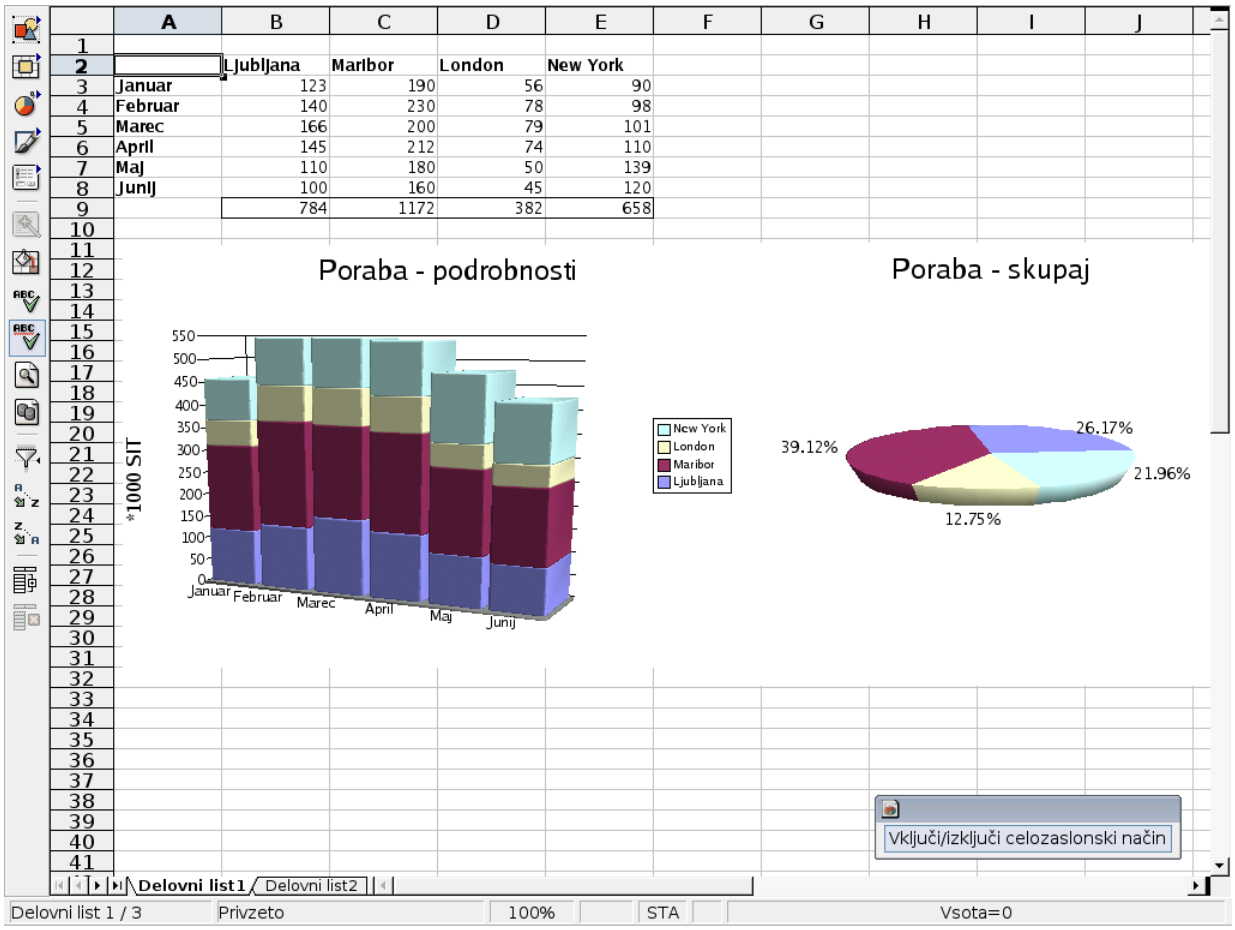
Slika 6.8: Določanje oznak

grafikon smo ustvarili iz celic `A2:E8`, skupno vsoto pa smo predstavili v posebnem grafikonu, ki predstavlja celice `B9:E9`.

6.1.6 Podatkovne zbirke v orodju OpenOffice.org Calc

Možnost dela s podatkovnimi zbirkami je že od nekdaj tudi del programov za delo s preglednicami. Kljub temu, da podatkovne zbirke največkrat obdelujemo s specializiranimi programi (npr. Access ali Paradox), pa lahko nekatere probleme zadovoljivo rešimo kar v preglednici. Pravzaprav imajo preglednice pred programi za delo s podatkovnimi zbirkami v nekaterih primerih celo prednost predvsem zaradi raznovrstne palete vgrajenih funkcij in lažje analize podatkov. Specializirani programi pa so seveda nujni pri večjih in kompleksnejših zbirkah, kjer potrebujemo veliko hitrost urejanja, iskanja in poizvedb. Ne smemo seveda pozabiti, da je pri preglednicah velikost ene tabele v podatkovni zbirki omejena z največjo velikostjo delovnega lista. Razlike pri delu s pravo podatkovno zbirko so tudi v (ne)možnosti povezovanja tabel.

Zbirke podatkov so (poenostavljeno) sestavljene iz zapisov (angl. *Records*), ki vsebujejo polja (angl. *Fields*). V preglednici lahko na primer kot zapise obravnavamo vrstice, kot polja pa stolpce oziroma celice s podatki. Zbirko podatkov lahko preberemo iz datoteke ali pa jo ustvarimo kar iz preglednice. OpenOffice.org Calc podpira delo tako z datotekami različnih programov za delo s podatkovnimi zbirkami, kot tudi s tekstovnimi datotekami.



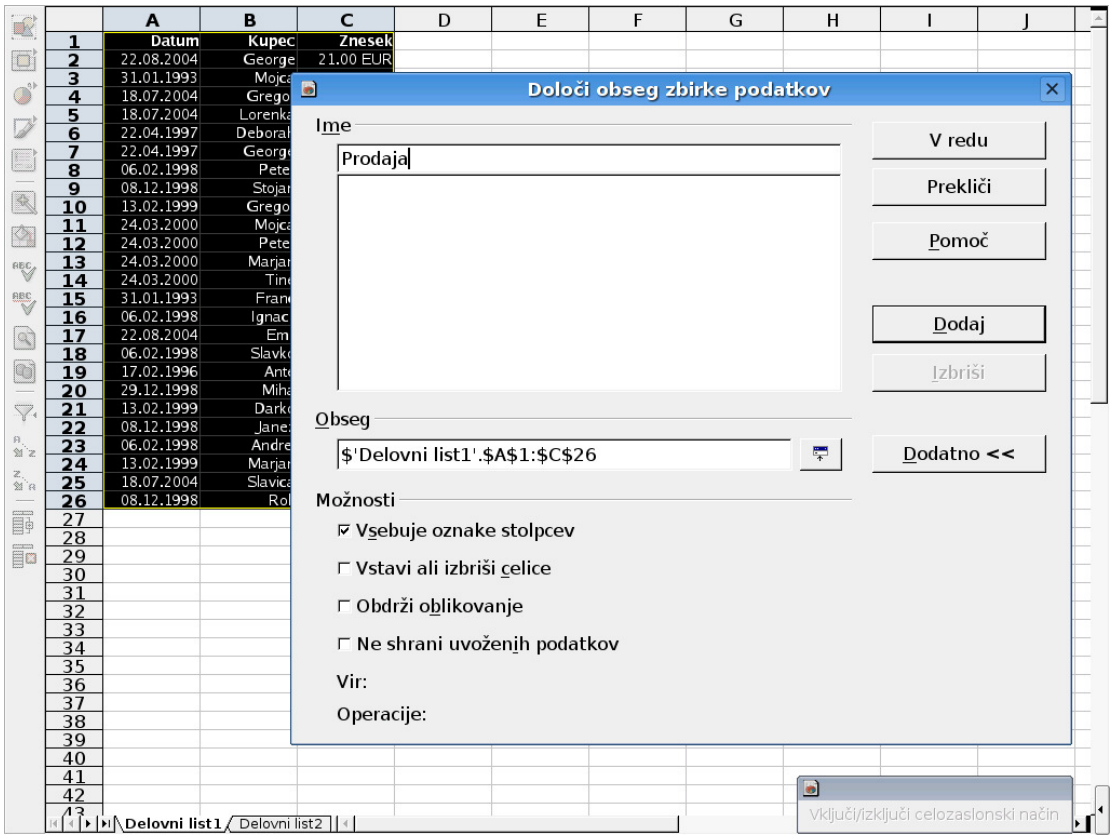
Slika 6.9: Tabela z grafikonoma

Ustvarjanje zbirke podatkov v preglednici

V prvo vrstico razpredelnice, ki bo predstavljala zbirko podatkov, vpišemo imena polj, na primer `Datum`, `Kupec` in `Znesek` (slika 6.10). Stolpce nato oblikujemo v skladu s podatki, na primer prvi stolpec kot datumski, drugega kot besedilo in tretjega kot valuta. V vsako naslednjo vrstico nato vpisujemo zapise.

Ko imamo podatke pripravljene, jih izberemo (lahko tudi cel delovni list) in z "Podatki→Določi obseg" definiramo podatkovno območje (slika 6.10). Vnesti moramo ime ter določiti, ali izbrano območje obsega tudi imena polj ("Vsebuje oznake stolpcev"). Po kliku na "V redu" je baza pripravljena.

Nad območjem, ki smo ga določili, lahko uporabljamo funkcije kot sta urejanje ali filtriranje, kot bi to počeli nad pravo bazo podatkov.



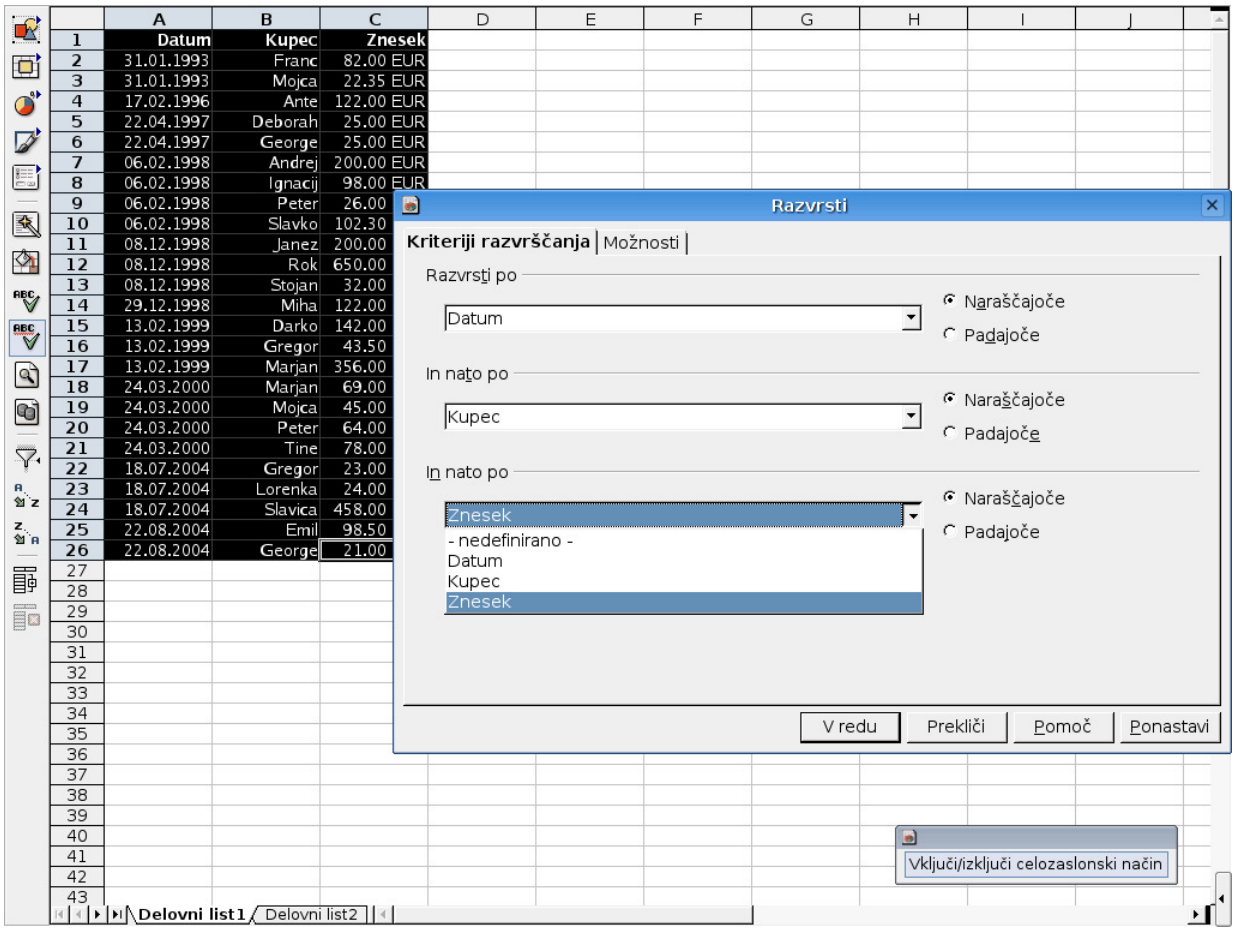
Slika 6.10: Ustvarjanje zbirke podatkov

Delo s podatki

Ko podatke definiramo kot bazo, jih lahko enostavno urejamo po kateremkoli polju (stolpcu). Primer na sliki 6.10 bi na primer želeli urediti najprej po datumih, nato pa še kupce po imenu in znesku (od najmanjšega do največjega).

Če imamo na delovnem listu več podatkovnih območij, moramo območje, s katerim bomo delali najprej izbrati v pogovornem oknu "Podatki→Izberi obseg", ali pa samo postavimo kurzor v pravo območje. Nato aktiviramo urejevalno okno z izbiro "Podatki→Razvrsti". Na sliki 6.11 vidimo, kako nastavimo iskalne kriterije.

Podatke lahko tudi filtriramo glede na določene kriterije. Tako lahko iz množice podatkov izluščimo tiste, ki jih potrebujemo. Enostavno filtriranje po vsebini, na primer prikaz zapisov samo za Gregorja, lahko izvedemo že z enostavnim filtrom, ki ga aktiviramo z "Podatki→Filtriraj→Samodejni filter" (spomnite se, da je ikona za hitri dostop do te funkcije tudi v glavni orodni vrstici). V zgornji vrstici tabele se pokažejo padajoči meniji, v katerih izberemo vrednost, na primer

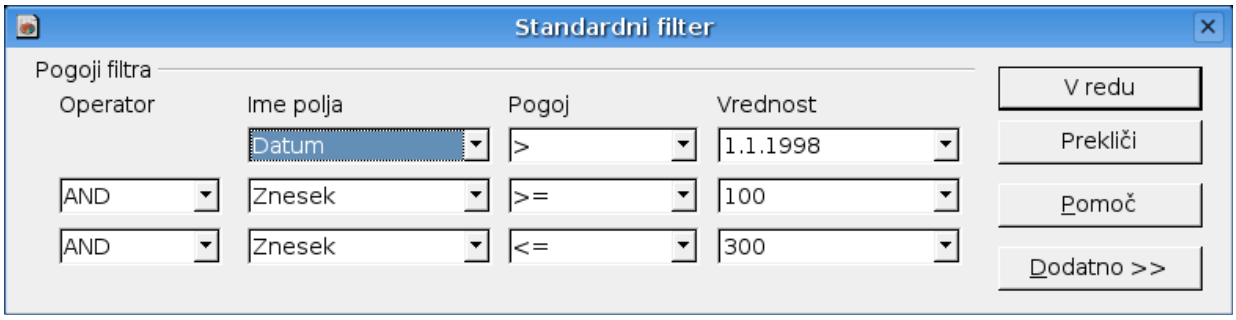


Slika 6.11: Urejanje po treh poljih

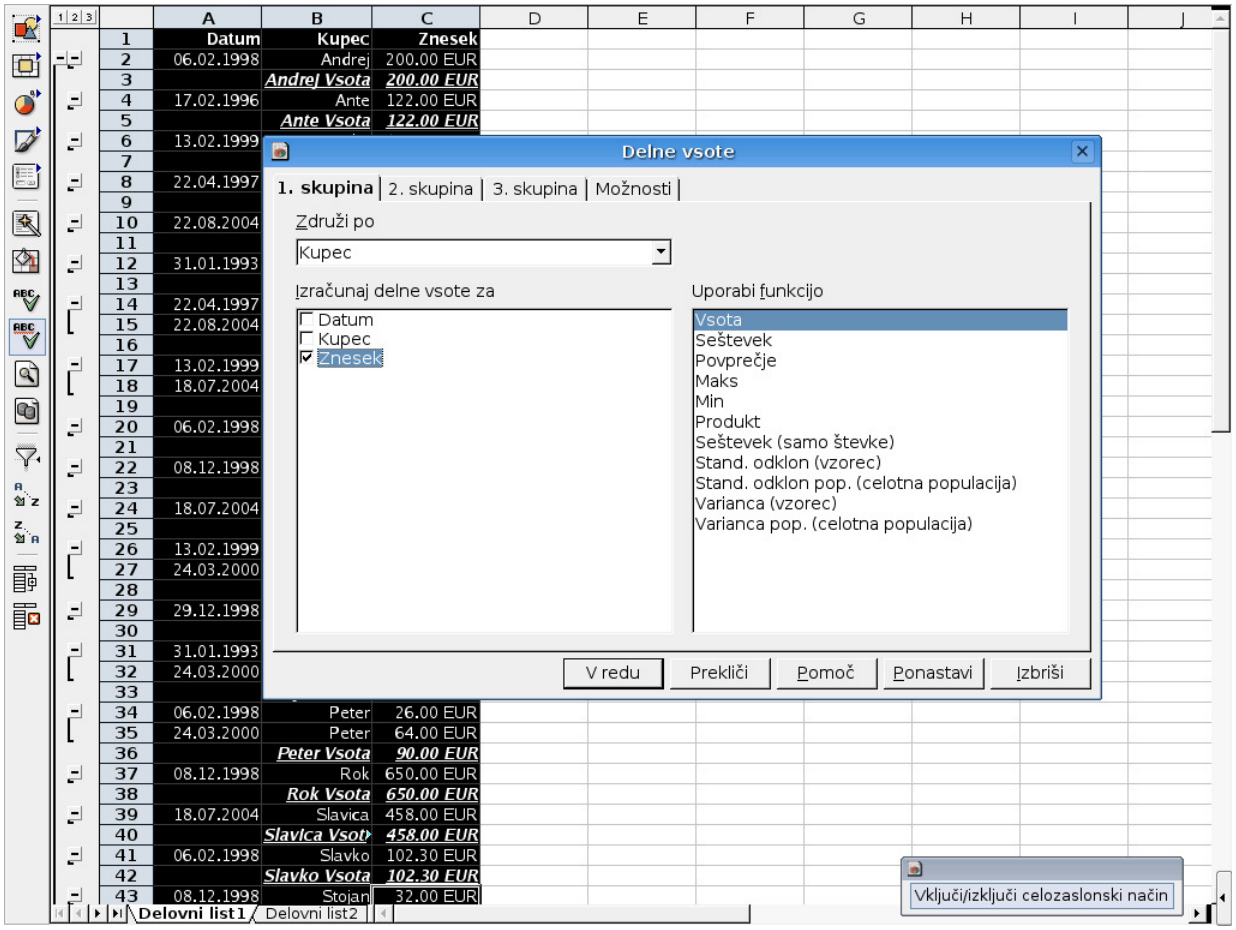
"Kupec→Gregor". OpenOffice.org Calc bo sedaj prikazal samo zapise, v katerih je kupec Gregor.

Največkrat pa so pogoji filtriranja precej bolj zapleteni. Take pogoje nastavljamo z izbiro "Podatki→Filtriraj→Standardni filter". V pogovornem oknu moramo določiti kriterij, ki ga sestavimo iz več pogojev. Pogoje povežemo z logičnima operatorjema `AND` ali `OR`, sestavimo pa jih iz imena polja ("Ime polja"), pogojnega operatorja ("Pogoj") in vrednosti ("Vrednost"). Pogojni operatorji so podobni relacijskim (tabela 6.2), dodani so še pogoji "največji", "najmanjši", "največji %" in "najmanjši %". Na sliki 6.12 vidimo pogoj, ki izbere tiste zapise o prodaji, ki vsebujejo podatke za transakcije po `1.1.1998`, njihova vrednost pa je bila med `100 EUR` in `300 EUR`.

Z orodjem "Podatki→Delne vsote" lahko našo tabelo zelo učinkovito analiziramo in ustvarjamo različne delne izračune, na primer vsote po dnevih, kupcih ip d. V pogovornem oknu (slika 6.13) vidimo, kako



Slika 6.12: Standardno filtriranje



Slika 6.13: Ustvarjanje delnih izračunov

nastavimo delne izračune vsote vseh transakcij za vsakega kupca posebej.

6.1.7 Shranjevanje preglednice

Kot v drugih modulih zbirke OpenOffice.org, lahko tudi s programom OpenOffice.org Calc shranjujemo dokumente v več formatih. Poleg formata `.sxc` lahko preglednico med drugim shranimo v formatu Microsoftovega programa Excel ali kot tekstovno datoteko, v kateri so polja med sebj ločena z vejicami (`.csv`). Za prikaz na spletu lahko preglednico pretvorimo tudi v jezik HTML.

6.1.8 Naloge

1. Kako se je imenovala in kaj je uporabniku ponujala prva resnično uspešna aplikacija za osebne računalnike?

2. Celico oblikujte tako, da se bo datum v njen zapisal v obliki: `Petek, 30.9.2004, 13:00`. Naslednjo celico pa oblikujte tako, da se bo vsako število v njej prikazalo kot ulomek.

3. Zakaj je priporočljivo, da ob delu z denarnimi zneski sami nastavimo oznako valute?

4. Zapišite primer in razlago vsebine celice, ki se sklicuje na celico v drugem delovnem listu!

5. Razložite razliko med absolutnim in relativnim naslavljanjem!

6. Na podlagi podatkov v celicah `F6:J15` na sliki 6.1 izračunajte v območju `F19:J22` maksimum, minimum, povprečje in standardno odstopanje (glejte funkcijo na sliki 6.2) posameznega stolpca! Pomagajte si z idejo relativnega naslavljanja.

7. V preglednici imamo naslednje podatke: v celici `A1` je število 2, v `A2` je 1, v `A3` je 3, v `B1` je 8, v `B2` je 5 in v `B3` je 4. V celico `C1` vpišemo formulo `=AVERAGE($A$1:B1)` in jo prekopiramo v celici `C2` in `D2`. Kakšen bo rezultat v celici `C2`? Kako bo zapisana formula v celici `D2`?

8. V celice `B1:F1` damo števila 1, 2, 3, 4 in 5. Podobno zapolnimo celice `A2:A6`. S pomočjo absolutnega in relativnega naslavljanja z eno samo formulo in uporabo kopiranja le-te izračunajte poštevanko nad podanimi števili!

9. Na XXVIII. poletnih olimpijskih igrah v Atenah 2004 so bili v finalu ženskega teka na 800 m doseženi naslednji rezultati: 1:59,62; 1:57,27; 1:56,38; 1:56,43; 1:56,51; 1:56,88; 1:56,43 in 2:00,95. S pomočjo absolutnega in relativnega naslavljanja z eno samo formulo in uporabo kopiranja le-te izračunajte zaostanke za najboljšim

rezultatom, nato pa tekmovalke razvrstite od prvega do osmega mesta! Kateri rezultat pripada Jolandi Čeplak ;-) ?

10. V celici `C9` imamo število 5. V `C10` zapišemo funkcijo:
`=IF(C9>10; C9; IF(C9>=0; "X!"; "Y!"))`
Kaj nam kot rezultat vrne ta funkcija? Razložite, zakaj!

11. Zakaj je vizualizacija podatkov zaželena?

12. Na podlagi podatkov v celicah `F6:J15` na sliki 6.1 prikažite v enem grafikonu uspeh po posameznih vprašanjih za vsakega študenta. Grafikon ustrezno opremite tudi z glavnim naslovom, naslovoma osi in legendo!

13. Kako lahko v preglednici predstavimo neko tabelo podatkovne zbirke? Omenili smo nekaj prednosti in pomanjkljivosti preglednic v primerjavi s programi za delo s podatkovnimi zbirkami. Kaj smo povedali?

14. Kako pretvorimo preglednico v jezik HTML?

6.2 Koristne spletne povezave

1. Spletna stran OpenOffice.org:
`http://www.openoffice.org/`

2. Spletna stran slovenskega OpenOffice.org:
`http://sl.openoffice.org/`
(S te strani je pod licenco GNU/FDL dostopna tudi knjiga [4].)

6.3 Literatura

[1] S. Haugland and F. Jones. *OpenOffice.org 1.0 Resource Kit.* Prentice-Hall PTR, Upper Saddle River, NJ, 2003.

[2] G. Leete, E. Finkelstein, and M. Leete. *OpenOffice.org for Dummies.* Wiley Publishing, Inc., Hoboken, NJ, 2003.

[3] R. Ludvik, I. Zajc, and A. Medic. *Hitri vodnik po OpenOffice.org.* Pasadena, Ljubljana, 2003. (Dostopna pod licenco GNU/FDL na: http://openoffice.lugos.si/knjiga/Hitri_vodnik_po_OpenOffice.org_FDL.pdf).

[4] E. R. Tufte. *The Visual Display of Quantitative Information.* Graphics Press, Cheshire, CT, 1983.

[5] E. R. Tufte. *Envisioning Information.* Graphics Press, Cheshire, CT, 1990.

[6] E. R. Tufte. *Visual Explanations.* Graphics Press, Cheshire, CT, 1997.

7

Računalniška grafika

Računalniška grafika obsega ustvarjanje raznovrstnih vizualnih informacij s pomočjo računalniške tehnologije. S pomočjo računalniške grafike je danes možno sintetizirati realistične upodobitve bodisi realnih ali povsem namišljenih predmetov in okolij. Ker ljudje zlahka in zelo hitro razumemo vizualne upodobitve [5, 6, 7], postaja računalniška grafika vse pomembnejše računalniško področje, ki se uporablja na vseh možnih področjih od uporabniških vmesnikov do skoraj vseh sodobnih uporabniških programov.

Grafične računalniške programe lahko ločimo po več kriterijih:

- glede na vrsto slik in objektov ločimo **2D** in **3D** računalniško grafiko,
- glede na to, ali gre za ustvarjanje posameznih slik ali za dinamično zaporedje slik, kot ga potrebujemo za animacijo, pa ločimo **statično** in **dinamično** računalniško grafiko
- glede na vrsto uporabniške interakcije ločimo **interaktivne** programe in take, predvsem računsko zahtevne, ki za podane vhodne parametre izračunajo zahtevano sliko v času, ki ga merimo v minutah, za zelo zahtevne upodobitve pa celo v urah.
- glede na vlogo slike ločimo grafične programe, kjer je **slika končni cilj**, kot na primer pri grafičnem oblikovanju, in programe, kjer je **slika le del nekega produkcijskega procesa**, kot so na primer modeli CAD (Computer Aided Design).
- glede na področje uporabe. Ker se dandanes računalniki uporabljajo praktično na vseh področjih človeškega delovanja, naštejmo le nekaj značilnih področij uporabe računalniške grafike:
  - računalniški uporabniški vmesniki,

- znanstvena vizualizacija in simulacija,
- inženirsko načrtovanje (CAD/CAM),
- kartografija in geografski informacijski sistemi (GIS),
- medicina,
- grafično oblikovanje in umetnost,
- animacija,
- navidezna resničnost (angl. *virtual reality*).

Glede na način generiranje slike na računalniškem monitorju ločimo vektorske in rastrske zaslone. Vektorski zasloni, ki so v preteklosti prvi omogočali prikaz vektorskih slik, oblikujejo sliko na zaslonu iz črt poljubne dolžine in naklona s pomočjo neposrednega krmiljenja elektronskega žarka v katodni cevi. Na rastrskem zaslonu pa je slika podobno kot na televizijskem ekranu sestavljena iz polja slikovnih pik (angl. *pixel*). Kvaliteta rastrske slike je odvisna od razsežnosti slikovnega polja (na primer 512×512, 1024×1024) in globine zapisa posameznega polja (ponavadi 8-bitni za črnobele zaslone in 24-bitni za barvne, po 8 bitov za rdečo, zeleno in modro barvo).

Glede na način zapisovanja grafičnih objektov ločimo **vektorsko** in **rastrsko** računalniško grafiko. Rastrska računalniška grafika (angl. *bitmap graphics*) obdeluje slike (na primer fotografije), ki so sestavljene iz pik, organiziranih v obliki 2D polja določene dimenzije. Če tako sliko ali njen izsek povečamo, postanejo posamezne pike vidne in zato postanejo objekti na sliki zrnati. Pri vektorski računalniški grafiki pa so posamezni elementi definirani v matematični obliki s pomočjo vnaprej določenih geometrijskih objektov (daljice, krogi elipse, pravokotniki, fonti, Bézierjeve krivulje ipd.) in so zato neodvisni od povečave. Tak zapis imajo tudi obrisno določene pisave. Vektorsko zapisane grafične objekte sicer tudi prikazujemo na rastrskih zaslonih, vendar se ti objekti vsakokrat prilagodijo najvišji ločljivosti izhodne naprave, bodisi računalniškega zaslona ali tiskalnika.

Pri grafični programski opremi ločimo naslednje koncepte:

Uporabniški grafični program, ki omogoča konstrukcijo ter spreminjanje in manipulacijo z grafičnimi objekti. Pri tem uporablja zmožnosti **grafičnega sistema**, ki igra funkcijo "kamere" in ki določa, kaj se vidi in kako se vidijo objekti na grafičnem zaslonu. Pri tem se spreminjajo medsebojna razmerja med koordinatnim sistemom objektov ter koordinatnim sistemom zaslona.

Grafična podatkovna struktura, ki predstavlja posamezne grafične elemente, ki sestavljajo na primer neko sliko, in njihove lastnosti (oblika, velikost, barva ipd.) ter njihova medsebojna razmerja (položaj, orientacija). V računalniški grafiki poznamo celo vrsto

standardnih grafičnih zapisov (png, gif, tiff, pict, jpeg, ps, eps, pdf za 2D podatke, vrml za 3D podatke, mpeg in avi za video posnetke), saj si želimo grafične zapise izmenjevati neodvisno od grafičnih programov, s katerimi so bili zapisi prvotno ustvarjeni.

Med grafičnimi formati velja posebej omeniti PostScript [1], saj je to pravzaprav programski jezik, ki je posebej namenjen vektorskemu opisu grafičnih elementov. Osnovna struktura jezika PostScript je sklad (angl. *stack*). Pri izpisu datoteke PostScript se pravzaprav interpretira program, ki je zapisan v datoteki PostScript. Ker so elementi PostScript zapisani vektorsko, se lahko poljubno povečajo ali pomanjšajo. Med drugim je PostScript najbolj razširjen format za zapis pisav. Format PostScript zapisa uporablja cela vrsta programov in tiskalnikov, bolj redko pa sami pišemo oziroma oblikujemo v PostScriptu.

Najbolj pomembne skupine uporabniških grafičnih programov so namenjene:

Barvanju in obdelavi fotografij. Ti programi uporabljajo rastrski zapis (na primer Adobe Photoshop in GIMP; slednjega bomo podrobneje spoznali v nadaljevanju poglavja). Tipične operacije, ki jih je možno izvajati nad rastrsko sliko, so: razne vrste filtriranja, kot so glajenje, izostritev in iskanje robov, ter spreminjanje barv in razne vrste preobrazb.

Risanju. Ti programi običajno uporabljajo vektorski zapis (na primer Adobe Illustrator in OpenOffice.org Draw; slednjega bomo spoznali v nadaljevanju poglavja). Tipične operacije nad vektorsko zapisanimi objekti so premik, rotacija, skaliranje, zrcaljenje, barvanje, senčenje ipd.

Načrtovanju. Ti programi so bodisi splošni 2D ali 3D načrtovalski programi, kot je na primer AutoCAD, ali pa so namenjeni določenemu uporabniškemu področju, kot na primer program ArchiCad za arhitekturo.

Izdelavi grafov in diagramov. Številni programi na osnovi numeričnih podatkov izdelajo različne vrste diagramov (točkovne, linijske, stolpične, tortne, polarne, kombinirane ipd.). To so lahko samostojni programi ali pa funkcije v sklopu drugih programov, kot so na primer OpenOffice.org Calc, Excel, Mathematica itd.

Izdelavi predstavitev. Za izdelavo prosojnic in računalniških predstavitev poznamo celo vrsto programov (na primer Microsoft PowerPoint in OpenOffice.org Impress; slednjega bomo podrobno spoznali v naslednjem poglavju).

3D animaciji, ki postaja za filmsko industrijo vse bolj pomembna.
Tipični programi za 3D animacijo so Maya, Lightwave in 3D Studio.

V nadaljevanju poglavja bosta predstavljena programa GIMP za delo z rastrskimi slikami in program OpenOffice.org Draw za delo z vektorsko grafiko. Omenili pa bomo tudi zanimiv program GraphViz.

7.1 GIMP

GIMP je program, ki je namenjen delu z bitnimi slikami. Ime GIMP je kratica za GNU Image Manipulation Program. Čeprav lahko tudi risbe, ustvarjene z modulom OpenOffice.org Draw, izvozimo kot bitne slike, pa GIMP omogoča določene posebne operacije, ki so značilne le za tovrstne programe. Ker gre za razmeroma preprost program, si bomo v tem poglavju ogledali le nekaj osnovnih načel za delo z bitnimi slikami.

Ko GIMP prvič poženemo, se pojavi pogovorno okno "GIMP User Installation", v katerem lahko določimo mapo namenjeno GIMP-u ("Personal GIMP Folder"), velikost vmesnega pomnilnika za slike ("Tile Cache Size") in pot do začasne mape ("Swap Folder"), kamor GIMP shranjuje dele slik, ki so prevelike za v pomnilnik. Dalje lahko nastavimo natančno ločljivost monitorja ("Monitor Resolution").

Ko zaključimo z nastavitvami, nas GIMP pozdravi z glavnim in pomožnim oknom, ki sta sestavljeni iz več razdelkov. V glavnem oknu so orodjarna, pod njo izbira barv in vzorcev, sledijo možnosti trenutno izbranega orodja in vrstica ikon za nadzor nad možnostmi. V pomožnem oknu so v dveh razdelkih (zgoraj in spodaj) zbrana številna pogovorna podokna, ki jih prikličemo s klikom na ustrezni zavihek. Obe okni sta plavajoči in poljubno nastavljivi. Obnašata se kot sidrišči za komponente. Posamezne komponente lahko odvlečemo na namizje in s tem ustvarimo novo plavajoče okno. Komponente lahko dodajamo, če kliknemo na ikono z majhno puščico levo in izberemo komponento v priročnem meniju "Add Tab".

Novo sliko ustvarimo iz menija z ukazom: "File→New" .

V pogovornem oknu *New Image* določimo ločljivost in tip nove slike. Ločljivost opredelimo s številom slikovnih elementov po horizontali in vertikali. Tip je lahko barvna slika (RGB) ali črno-bela slika (Grayscale). V GIMP-u sicer lahko obdelujemo tudi slike s paleto (Indexed), če jih kasneje pretvorimo ali preberemo iz datoteke (npr. GIF). Barvne slike imajo trikrat večjo bitno globino kot črno-bele, ker porabijo za zapis vsake točke tri barvne komponente. Večji kot sta ločljivost in globina slike, večji bosta njena kakovost in količina zasedenega pomnilnika.

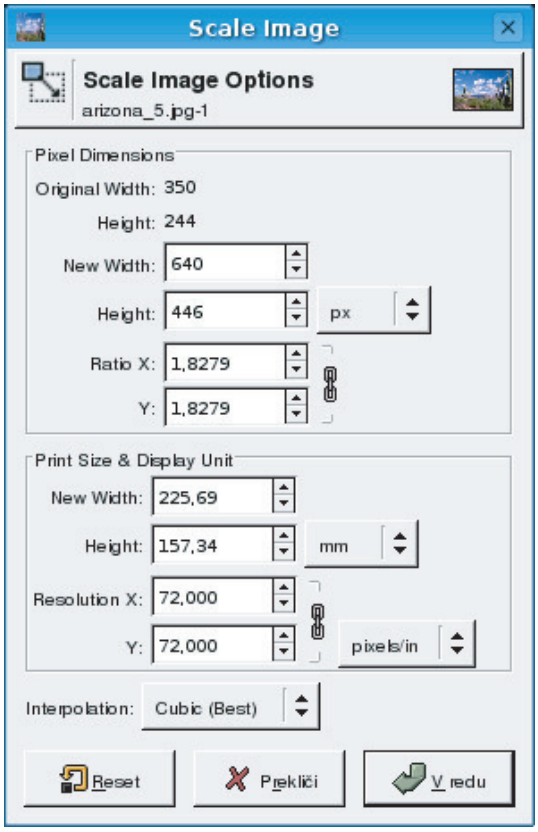
Ko z gumbom "OK" potrdimo izbrano, se pojavi novo okno s prikazom

slike. Naenkrat lahko obdelujemo več slik in vsaka se bo prikazovala v svojem oknu. Sliko (angl. *image*) shranimo s "File→Save", s "File→Save As" pa jo preimenujemo oziroma shranimo v poljubnem formatu. GIMP-ov lastni format je XCF.

7.1.1 Velikost, barvna globina in plasti

Oglejmo si, kako lahko slikam spreminjamo ločljivost in barvno globino ter kako si lahko pri obdelovanju slik pomagamo s plastmi.

Spreminjanje velikosti



Slika 7.1: Spreminjanje velikosti slike

Velikost slike določimo že na začetku, ko jo ustvarimo. Z ukazom "Image→Scale Image" jo lahko naknadno spremenimo. Odpre se pogovorno okno, ki je prikazano na sliki 7.1. V poljih "Width" in "Height" nastavimo širino in višino slike. To lahko storimo v različnih enotah, najlažje v številu slikovnih elementov ("px") ali relativno glede na trenutno velikost (v %). Če označimo ikono "Constant aspect ratio"

(desno od polj "Ratio"), se bo pri spreminjanju višine samodejo popravila še širina slike, tako da bo razmerje med njima ostalo enako.

Z "Interpolation" določimo, na kakšen način se izračunajo vrednosti novih slikovnih elementov: brez ("None"), z linearno ("Linear") ali s kubično ("Cubic") interpolacijo. Najboljši rezultat običajno dobimo s kubično interpolacijo.

Spreminjanje barvne globine

V podmeniju "Image→Mode" lahko spremenimo barvno globino. Način "RGB" pomeni običajno 24-bitno barvno sliko, "Grayscale" pomeni 8-bitno sivinsko sliko in "Indexed" pomeni sliko, v kateri lahko vsaka točka zavzame zgolj eno od barv iz končne palete (s tem načinom varčujemo s pomnilnikom pri slikah, ki vsebujejo zelo malo različnih barv ali odtenkov, saj je za zapis posamezne točke potrebno shraniti samo dovolj bitov za zapis zaporednega mesta v paleti).

Številna orodja in učinki ne delujejo na slikah s paleto (Indexed), zato jih je potrebno pretvoriti v barvne (RGB). Če želimo privarčevati pri velikosti shranjene slike (npr. če izdelujemo elemente spletnih strani), najprej obdelamo sliko v polni barvni globini, nato pa običajno izberemo paleto z majhnim a še zadostnim številom barv in sliko shranimo v obliki, ki je primerna za slike s paleto (PNG ali GIF).

Plasti

Slika, ki jo obdelujemo, je lahko sestavljena iz več plasti (angl. *layers*), od katerih je vsaka sestavljena iz enega ali večih barvnih kanalov. Sliko v GIMP-u si najlaže predstavljamo kot skladovnico prosojnic, kjer je končni rezultat pogled skozi vse prosojnice naenkrat. Vsaka plast ima poleg barve v vsaki točki določeno stopnjo prosojnosti, zato imamo poleg kanalov za rdečo, zeleno in modro barvo še tako imenovani kanal alfa (angl. *alpha channel*), ki vsebuje podatke o prosojnosti. Edino plast popolnoma v ozadju je lahko brez kanala alfa. Bolj zahtevni uporabniki lahko nastavijo še druge lastnosti, ki določajo, kako se barvne plasti med sabo združujejo za različne vizualne učinke.

Plasti, iz katerih je sestavljena slika, so prikazane znotraj pomožnega okna v pogovornem podoknu "Layers", ki ga lahko prikažemo tudi z ukazom "Dialog→Layers". S klikom na ime plasti izberemo aktivno plast. Če pritisnemo na ikono z očesom, lahko posamezno plast začasno skrijemo. Ikone pod seznamom plasti predstavljajo ukaze za dodajanje plasti, premikanje v ospredje ali ozadje, podvajanje, lepljenje in brisanje. Vsi ukazi, ki jih lahko izvedemo nad plastmi, so navedeni v meniju "Layer". Novo plast ustvarimo z ukazom "Layer→New Layer". Kopijo izbrane

plasti ustvarimo z ukazom “Layer→Duplicate Layer”.

Večina operacij nad slikami se izvede nad trenutno izbrano plastjo. Zapomnimo si, da nekatere operacije, kot je naprimer premikanje ali lepljenje (*Paste*) izbranega območja, ustvarijo začasno plast, ki obstaja, dokler operacije ne zaključimo ali izvedemo ukaza “Layer→Anchor Layer”, ki nalepi začasno plast na plast pod njo.

7.1.2 Osnovna slikarska orodja

V glavni orodjarni so osnovna orodja za delo s sliko. Predstavljena so v tabeli 7.1. Orodje izberemo s pritiskom na ikono. Ko orodje izberemo, se v razdelku “Tool Options”, ki se nahaja pod glavno orodjarno, prikažejo možnosti orodja, ki jih lahko spreminjamo. Nekatere možnosti orodij (oblika čopiča, vzorec in gradient) lahko spreminjamo tudi v spodnjem razdelku pomožnega okna. Orodje ostane izbrano, dokler ne izberemo naslednjega. Še več orodij najdemo v meniju “Tools”, ki se nahaja nad sliko. Do menijev lahko pridemo tudi z desnim klikom na sliki.

Povečava in pomanjšava

Orodje *Magnify* omogoča pomanjšavo in povečavo prikaza slike. Pritisk na sliko jo poveča, če pa pridržimo `<Ctrl>`, pritisk sliko pomanjša. Stopnjo povečanja lahko določimo tudi z “View→Zoom” iz menija. Z ukazom “View→Zoom→Other” lahko stopnjo povečanja določimo neposredno. V polje “Zoom” vpišemo stopnjo povečanja (do 25600%) ali stopnjo pomanjšanja (vrednosti manjše od 100%). Velike povečave potrebujemo, ko želimo veliko natančnost posegov v sliko (na primer spreminjanje vrednosti posameznega slikovnega elementa). Stopnjo lahko določimo tudi kot razmerje (Zoom Ratio).

Označevanje

Pravokotno območje slikovnih elementov označimo z orodjem *Rect Select*. Kliknemo na skrajnem robu območja, ki ga želimo označiti, in povlečemo miško v diagonalno nasprotni rob, kjer gumb spustimo. Z orodji *Ellipse Select*, *Free Select*, *Fuzzy Select*, *Color Select* in *Scissors* lahko na različne načine označimo območja poljubnih oblik. Z uporabo možnosti “Mode” v pogovornem oknu z možnostmi orodja lahko izberemo logično operacijo nad območji, ki jih označujemo. Območja lahko nadomeščamo (“Replace”), seštevamo (“Add”), odštevamo (“Subtract”) in tvorimo preseke (“Intersection”). Lahko izberemo tudi glajenje robov (“Antialiasing”) in mehčanje robnega območja (“Feather”).

Z označevanjem v GIMP-u določamo območje slikovnih elementov, nad

Ikona	*Ime orodja*	*Opis orodja*
	Rect Select	Označevanje pravokotnih regij
	Ellipse Select	Označevanje eliptičnih regij
	Free Select	Označevanje ročno narisanih regij
	Fuzzy Select	Označevanje zveznih regij
	Color Select	Označevanje regij po barvi
	Scissors	Označevanje oblik na sliki
	Paths	Ustvarjanje in urejanje poti (krivulj)
	Color Picker	Izbiranje barv iz slike
	Magnify	Povečava in pomanjšava
	Measure	Merjenje razdalj in kotov
	Move	Premikanje plasti ali izbire
	Crop/Resize	Obreži ali spremeni velikost slike
	Rotate	Vrtenje plasti ali izbire
	Scale	Skaliranje plasti ali izbire
	Shear	Nagibanje plasti ali izbire
	Perspective	Spreminjanje perspektive plasti ali izbire
	Flip	Prevračanje plasti ali izbire
	Text	Dodaj besedilo sliki
	Bucket Fill	Zapolni z barvo ali vzorcem
	Blend	Zapolni s prehajanjem barv
	Pencil	Riši točke z ostrim robom
	Paintbrush	Riši medle poteze s čopičem
	Eraser	Zbriši do ozadja ali prosojnosti
	Airbursh	Riši kot z razpršilcem
	Ink	Riši s črnilom
	Clone	Riši z uporabo vzorcev ali delov slik
	Convolve	Zmehčaj ali izostri
	Smudge	Vleči barvo po sliki
	Dodge/Burn	Svetljenje in temnjenje

Tabela 7.1: Orodja v glavni orodjarni programa GIMP

Slika 7.2: Označevanje pravokotnega območja

katerimi kasneje izvajamo določene operacije. Pravzaprav za vsako točko določimo stopnjo vpliva nadaljnih operacij nad območjem, s čimer lahko dosežemo mehke prehode ob robovih. Z ukazom "Select→Toggle QuickMask" (ali s pritiskom `<Shift>+<Q>` ali s klikom na ikono s črtkastim pravokotnikom spodaj levo od slike) lahko izberemo način, kjer je trenutna izbira prikazana kot maska. Neoznačen del slike je obarvan pordečeno, prosojni deli pa označujejo izbrano območje. Vmesni odtenki prikazujejo vmesne stopnje. Po maski lahko rišemo s poljubnimi orodji, ko pa ponovno kliknemo na ikono spodaj levo, jo pretvorimo nazaj v izbiro, nad katero lahko izvajamo nadaljnje operacije. Na sliki 7.2 vidimo označeno pravokotno območje, masko območja in masko istega območja, če uporabimo mehčanje (Feather).

Pozor! Če po maski ne moremo risati, je vzrok verjetno v tem, da smo kakšno območje premaknili in ga pozabili prilepiti, zato je še vedno aktivna začasna plast premaknjenega območja. Začasno plast moramo najprej prilepiti z ukazom "Layer→Anchor Layer" (ali s pritiskom na ikono s sidrom).

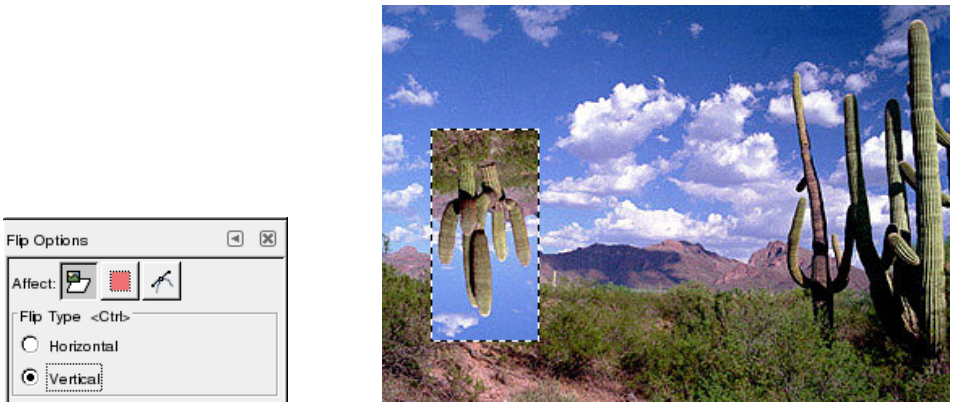
Pri označevanju si lahko pomagamo tudi z mrežo, ki jo postavimo na sliko z ukazom "View→Show Grid". Gostoto in način prikaza mreže določimo v pogovornem oknu, ki ga dosežemo z ukazom "Image→Configure Grid". Mreža ni del slike, zato se ne natisne in ne shrani.

Osnovne operacije

Osnovne operacije nad celo sliko so zbrane v meniju "Image". Celotno sliko lahko obračamo z ukazom "Image→Transform→Rotate" in prevračamo z "Image→Transform→Flip". Sliko lahko obrežemo z ukazom "Image→Crop Image" ali samodejno z "Image→Transform→Autocrop".

Spreminjanje velikosti in obračanje lahko izvajamo le na celotni sliki, prevračamo pa lahko tudi izbrano območje z orodjem *Flip*. Izrezovanje uporabimo, ko želimo izbrani del slike shraniti kot novo sliko. Primer prevračanja območja je na sliki 7.3.

Z orodjema *Pencil* in *Paintbrush* lahko po sliki prostoročno rišemo. Razlika je v tem, da *Pencil* pušča grobe robove, *Paitbrush* pa jih mehča.



Slika 7.3: Prevračanje območja

Možnosti risarskega orodja spreminjamo v pogovornem oknu pod glavno orodjarno. Barvo, obliko čopiča (*Brush*), vzorec (*Pattern*) in gradient (*Gradient*) izberemo kar s pritiskom na ikono, lahko pa jih najdemo tudi v komponentah, ki se nahajajo v pomožnem oknu. Pod glavno orodjarno lahko nastavimo še stopnjo prosojnosti (*Opacitiy*), kjer 0,0 pomeni popolnoma prosojno in 100,0 popolnoma neprosojno. *Mode* določa učinek risanja. *Fade Out* povzroči da risanje po določeni razdalji zbledi. Z *Use Color from Gradient* uporabimo za risanje barve izbranega gradienta.

Slika 7.4: Uporaba kaligrafskega čopiča, pisala, črnila in razpršilca

Podobno uporabljamo orodje *Airbrush*, ki posnema nanašanje barve z razpršilcem. Orodje ima dodatni nastavitvi *Rate* in *Pressure*, ki posnemata parametre razpršilca.

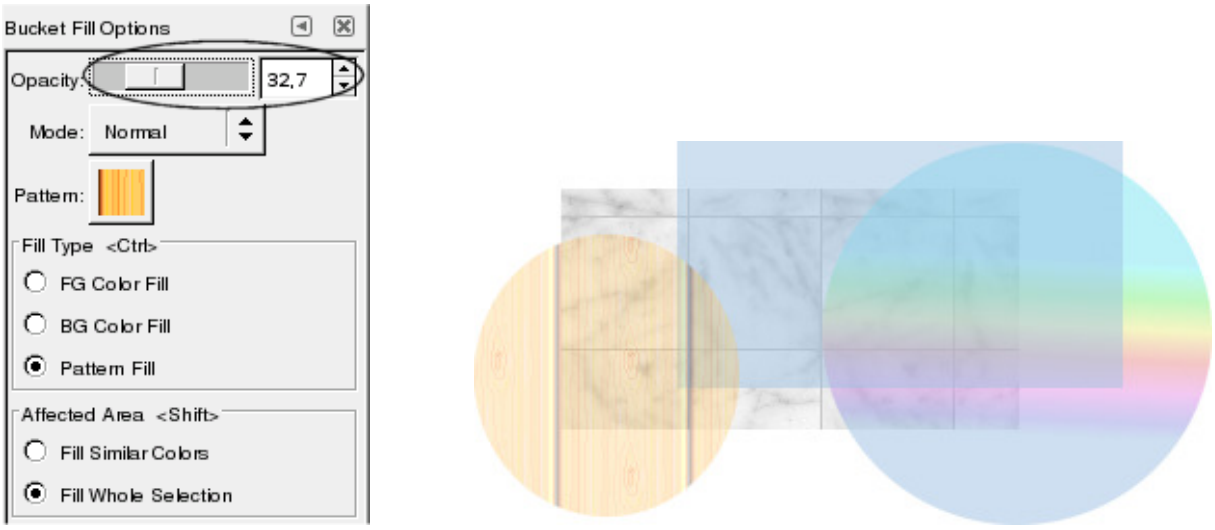
Orodje *Ink* oponaša risanje s črnilom. Lahko nastavimo tip čopiča (*Type*), velikost (*Size*), kot (*Angle*), ter parametre občutljivosti (*Sensitivity*), ki so velikost (*Size*), nagib (*Tilt*) in hitrost (*Speed*).

Orodje *Clone* je podobno kot žig, saj lahko rišemo kar z vzorci ali deli slik.

Če nastavimo pisalu barvo ozadja (na primer belo), lahko z njim tudi brišemo. Orodje *Eraser* pa oponaša radirko. Z orodjem *Convolve* lahko

sliki spreminjamo ostrino. Z orodjem *Smudge* lahko barvo "razmažemo" po sliki. Z orodjem *Dodge/Burn* pa lahko sliko na mestih potemnimo ali posvetlimo.

Primer uporabe risarskih orodij je na sliki 7.4. Primer prepuščanja barve ozadja pa vidimo na sliki 7.5.



Slika 7.5: Uravnavanje prosojnosti

Ker GIMP ni namenjen vektorskemu risanju, nimamo orodij za risanje matematičnih likov. Lahko pa like rišemo tako, da označimo območje in ga zapolnimo z orodjema *Bucket Fill* ali *Blend*. *Bucket Fill* nam omogoča polnjenje z barvo ali vzorcem. Če izberemo možnost "Fill Whole Selection", lahko pobarvamo celotno izbrano območje. Zanimivo pa je orodje *Blend*, s katerim narišemo barvni prehod (gradient). Prehod narišemo tako, da pritisnemo na gumb, povlečemo miško, da določimo smer in velikost prehoda, ter spustimo gumb. Če pritisnemo ikono "Gradient", lahko izberemo že pripravljene barvne prehode (npr. mavrične barve).

Na voljo imamo še nekaj orodij za geometrične transformacije izbranega območja ali plasti. Z *Rotate* območje zavrtimo. S *Scale* mu spremenimo velikost. S *Shear* območje zamaknemo, kot bi bilo nagnjeno. S *Perspective* oponašamo učinek perspektive.

Če med ustvarjanjem slike izvedemo nezaželeno operacijo, lahko sliko vedno povrnemo v prejšnje stanje z ukazom "Edit→Undo". Preklicano operacijo ponovimo z ukazom "Edit→Redo". GIMP si zapomni zgodovino operacij (če je na voljo dovolj pomnilnika), ki si jo lahko ogledamo v pogovornem oknu *Undo history*, ki se nahaja v pomožnem oknu, lahko pa jo prikličemo tudi z ukazom "Edit→Undo History". V pogovornem oknu je vsak korak prikazan grafično in opisno. Lahko izberemo poljubno točko

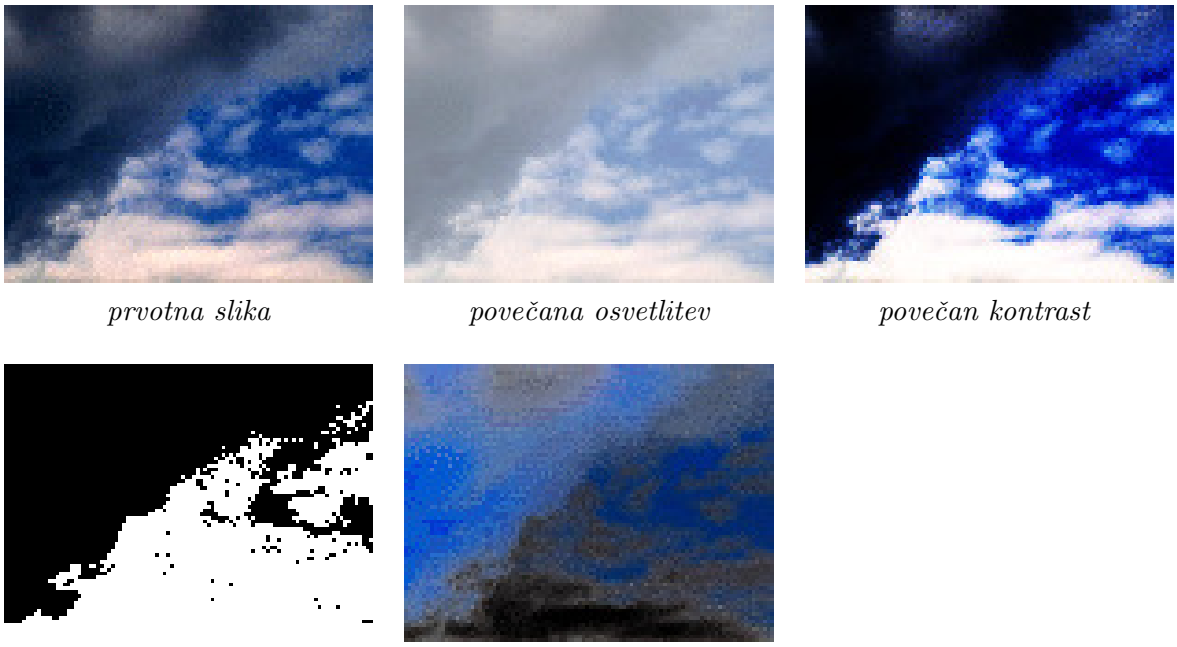
v zgodovini, vendar bodo nadaljne spremembe iz zgodovine zbrisale ukaze, ki smo jih naredili za tem in jih nadomestile z novimi!

7.1.3 Spreminjanje barve

Orodja in ukazi za uravnavanje barv (barvni filtri) se nahajajo v meniju "Tools→Color Tools". Vse operacije lahko izvedemo na celotni sliki ali pa na predhodno izbranem območju (orodja *Select*).

Kontrast in osvetlitev

Najpreprostejše uravnavanje je spreminjanje osvetljenosti in kontrasta slike. Spreminjamo ju lahko v pogovornem oknu, ki ga dosežemo z ukazom "Tools→Color Tools→Brightness-Contrast". Na sliki 7.6 vidimo fotografijo, kateri smo v prvem primeru izrazito povečali osvetlitev, na desni sliki pa smo povečali kontrast.



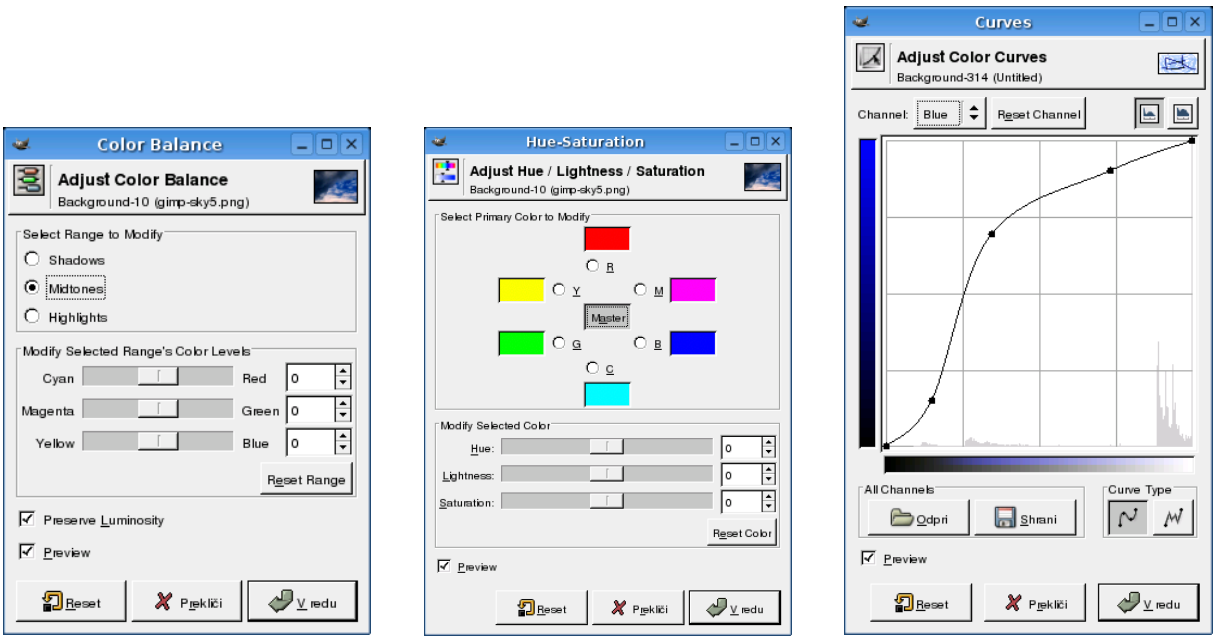
prvotna slika *povečana osvetlitev* *povečan kontrast*

črno-bela slika *inverzna slika*

Slika 7.6: Spreminjanje barve

Barvni filtri

Z ukazom "Tools→Color Tools→Color Balance" iz menija prikličemo pogovorno okno, ki je prikazano na sliki 7.7 levo. Z drsniki za uravnavanje barve lahko sliki odvzamemo ali dodamo posamezno barvno komponento.



Slika 7.7: Uravnavanje barv na sliki

Negativni deleži barve, izraženi v odstotkih, predstavljajo odvzemanje posamezne barve, pozitivni pa dodajanje. Parametre lahko določamo posebej za srednje tone (*Midtones*), temne tone (*Shadows*) in svetle tone (*Highlights*). Z možnostjo "Preserve Luminosity" ohranimo predhodno svetlost točk. Z "OK" se spremembe izvedejo na vsakem izbranem slikovnem elementu posebej.

Podobno deluje ukaz "Tools→Color Tools→Hue-Saturation", s katerim lahko v pogovornem oknu, ki je prikazano na sredini slike 7.7, spreminjamo barvni ton (*Hue*), svetlost (*Lightness*) in zasičenost (*Saturation*) vsaki barvni komponenti posebej. Barvno komponento, ki jo spreminjamo, ocnačimo v zgornjem delu pogovornega okna.

Z ukazom "Tools→Color Tools→Curves", pa lahko barvne komponente spreminjamo tako, da oblikujemo krivuljo, ki opisuje preslikavo med staro in novo vrednostjo. Primer je na sliki 7.7 desno.

Odprimo katerokoli barvno sliko, na primer `bigapple.gif`, ki se nahaja v imeniku `/usr/lib/openoffice/share/gallery`. Z ukazom "Image→Mode→RGB" jo pretvorimo v 24-bitne barve. S "Tools→Color Tools→Color Balance" iz menija povečajmo vsebovanost rdeče barve za 90%, zeleno pa zmanjšajmo za isto vrednost. Jabolko je postalo na pogled bistveno bolj zrelo.

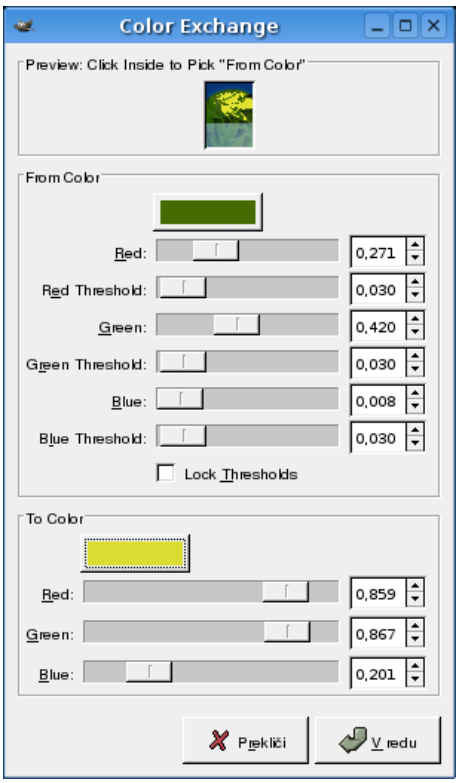
Z ukazom "Image→Mode→Grayscale" pretvorimo sliko v sivinsko. Če želimo pretvoriti sliko v črno-belo (binarno) sliko, uporabimo ukaz

"Tools→Color Tools→Threshold". Z miško ali z vnosom mejnih vrednosti lahko določimo območje sivinskih vrednosti, ki se bodo pretvorile v belo, ostale vrednosti pa se bodo pretvorile v črno.

Zanimiv učinek dobimo z uporabo ukaza "Tools→Color Tools→Posterize", s katerim zmanjšamo število barvnih nivojev in slika postane podobna postru. Barve pa lahko invertiramo z že omenjenim ukazom "Layer→Colors→Invert".

Učinek opisanih orodij je prikazan na sliki 7.6.

Menjava barve



Slika 7.8: Orodje *Color Exchange*

Če želimo zamenjati neko barvo na sliki z drugo, lahko to storimo z orodjem *Color Exchange* (menjava barve). Najdemo ga v meniju "Filters→Colors→Map→Color Exchange". Pojavi se pogovorno okno *Color Exchange*, ki se nahaja na sliki 7.8. V oddelku "From Color" določimo barvo, ki jo želimo zamenjati. Barvo lahko izberemo kar z miško v oddelku "Preview". Pomaga, če pred uporabo orodja izberemo približno območje, kjer se barva nahaja, saj se nam potem del izbranega območje prikaže kot "Preview". V vrsticah lahko natančno nastavimo vrednost

posameznih barvnih komponent in toleranco (angl. *Threshold*), koliko odstopanja od barve še dovolimo. Če pritisnemo na ikono z barvo, se prikaže standardno pogovorno okno za izbiro barve. V oddelku "To Color" določimo novo barvo, s katero bomo zamenjali staro. Kako bo izgledala menava, vidimo v oddelku "Preview". Ko smo zadovoljni, pritisnemo "V redu" in zamenjava se bo izvedla na celotnem izbranem območju.

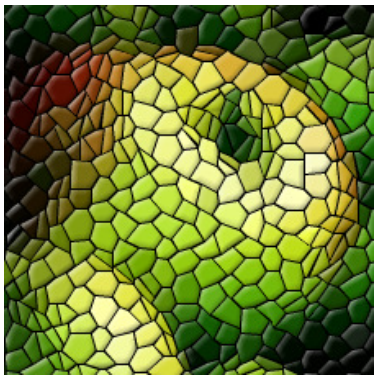
7.1.4 Učinki

Ena od najbolj značilnih operacij nad slikami je dodajanje učinkov (angl. *effects*). Za njimi se skrivajo bolj ali manj zapletene matematične operacije, s katerimi poskušamo spremeniti videz slike. Za uspešno uporabo ni potrebno natančno razumevanje matematičnih operacij, temveč predvsem izkušnje in nekaj umetniške žilice.

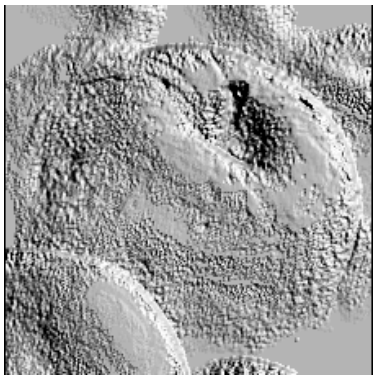
Različne učinke uporabljamo na enak način. Izvajamo jih lahko na celotni sliki, na izbrani plasti ali pa na predhodno izbranem območju. Zbrani so v meniju "Filters". Večina učinkov prikaže pogovorno okno, v katerem prilagajamo njihove parametre. V nadaljevanju bodo na kratko predstavljeni preprosti učinki, ki jih ponuja GIMP. Podobni učinki so zbrani v skupinah.

- **Blur** pomeni zmanjšanje ostrine slike. To lahko storimo na več načinov, od katerih se najpogosteje uporablja Gaussovo glajenje (*Gaussian Blur*). Če želimo doseči učinek gibanja, pa uporabimo *Motion Blur*.
- **Colors** je zbirka učinkov, ki delujejo na barvah. S *Colorify* lahko obarvamo sivinske slike. Z *Value Invert* lahko dosežemo učineg negativa.
- **Noise** pomeni šum, s čimer običajno naredimo sliko bolj zrnato. Z *Noisify* lahko dodamo šum posamično vsaki komponenti. S *Spread* naključno razmečemo točke, kar ostro sliko naredi tako, kot bi jo narisali s kredo ali ogljem.
- **Edge Detect** pomeni detekcija robov. Lahko izberemo številne postopke za odkrivanje robov, od katerih sta pogosto uporabljana *Laplace* in *Sobel*.
- **Enhance** pomeni izostritev slike. S *Sharpen* izostrimo robove na sliki. Z *Deinterlace* odpravimo prepletenost, če smo sliko zajeli iz televizije. Z *Despeckle* odpravimo zrnatost pri slikah pridobljenih z optičnim čitalcem.
- **Generic** so splošna orodja. Učinek *Dilate* je podoben raztapljanju, *Erode* pa je nasproten učinek podoben eroziji.

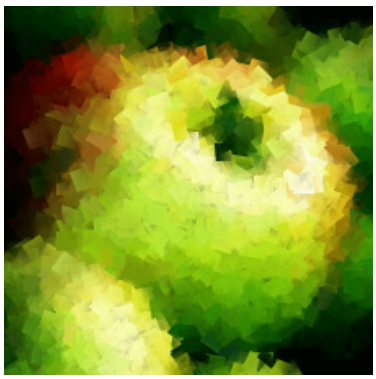
prvotna slika



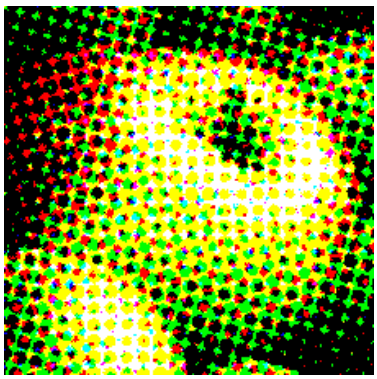
Mosaic



Emboss



Cubism



Newsprint

Ripple

Slika 7.9: Učinki paketa GIMP

- **Glass Effects** so učinki stekla. *Apply Lens* oponaša lečo, *Glass Tile* pa steklo s ploščičastim vzorcem.

- **Light Effects** so svetlobni učinki. *FlareFX* in *GFlare* oponašata učinek, ko sonce sveti v kamero. *Sparkle* oponaša iskrico. *Supernova* prikaže svetleče območje, iz katerga izhajajo žarki.

- **Distorts** so učinki, ki na različne načine "pokvarijo" sliko. *Emboss* naredi sliko podobno reliefu. *Mosaic* sliko pretvori v mozaik. *Newsprint* oponaša grobe tiskarske vzorce. *Ripple* in *Waves* oponašata valove. *Video* oponaša prikaz na video monitorju. *Wind* oponaša veter.

- **Artistic** so umetniški učinki. Z *Apply Canvas* lahko sliko narišemo na "platno". *Cubism* nariše sliko iz kock.

- **Map** so razni učinki, ki oponašajo mečkanje in trganje. *Bump Map* doda sliki navidezne izbokline.

- **Render** pomeni izrisovanje. To pravzaprav niso učinki, ampak so podporgrami, ki rišejo različne vzorce, kot so oblaki (*Clouds*), šahovnice (*Checkerboard*), labirinti (*Maze*) in fraktali (*Fractal Explorer*).

Primeri nekaterih učinkov se nahajajo na sliki 7.9.

7.1.5 Napredne možnosti programa GIMP

Ker je GIMP program z odprto kodo, vsaka nova različica prinaša številne nove funkcije in možnosti, ki jih lahko spreminjamo pri obstoječih orodjih. V tem poglavju smo predvsem preleteli osnovne operacije, ki jih moramo poznati, če želimo računalniško spreminjati slike. Skoraj vsa orodja pa imajo bistveno več možnosti, kot smo jih navedli in za učinkovito uporabo je potrebno nekaj prakse in eksperimentiranja.

Poleg vgrajenih funkcij pa GIMP ponuja tudi razširitve v obliki dodatnih komponent (angl. *plug-in*) in skript. Do njih pridemo prek menija "Xtns" v glavnem oknu ali prek menija "Script-Fu". Tu so najrazličnejši predprogramirani pripomočki, učinki, generatorji gumbov in animacij itd. GIMP vsebuje svoj skriptni jezik, z uporabo katerega lahko obdelujemo slike tudi neinteraktivno.

7.2 OpenOffice.org Draw

OpenOffice.org Draw je modul paketa OpenOffice.org, ki ga uporabljamo za delo z vektorsko grafiko.

7.2.1 Osnovna risarska orodja

OpenOffice.org Draw nudi veliko možnosti za delo z vektorsko grafiko. V tem poglavju si bomo pogledali najosnovnejše.

Novo risbo ustvarimo iz menija z ukazom
"Datoteka→Nov→Drawing"
Pojavijo se prazna risba, ki zavzema največji del ekrana, in orodne vrstice. Na vrhu sta funkcijska in predmetna orodjarna, na levi strani ekrana pa se nahaja glavna orodjarna. Prikaz posameznih orodnih vrstic vklapljamo in izklapljamo v "Pogled→Orodne vrstice".

Risbo (angl. *drawing*) shranimo s "Datoteka→Shrani" ali preimenujemo z ukazom "Datoteka→Shrani kot". Z ukazom "Datoteka→Izvoz" jo lahko izvozimo v bitno grafiko in jo naprej obdelujemo kot bitno sliko. Lahko jo izvozimo tudi kot dokument HTML, Adobe PDF ali Macromedia Flash.

Ikona	*Ime orodja*	*Opis*
	*Izberi*	Označevanje objekta ali skupine objektov
	*Zoom*	Povečava in pomanjšava prikaza risbe
	*Besedilo*	Oblikovanje besedila v risbi
	*Pravokotnik*	Risanje pravokotnikov
	*Elipsa*	Risanje elips, krogov, krožnih izsekov
	*3-D predmeti*	Risanje 3D objektov
	*Krivulja*	Risanje krivulj, poligonov
	*Črte in puščice*	Risanje črt in puščic
	*Konektor*	Risanje povezav
	*Učinki*	Različni učinki (rotacija, nagnjenost ipd.)
	*Poravnava*	Poravnava objektov
	*Razporedi*	Razvrščanje objektov
	*Vstavi*	Vstavljanje objektov iz drugih aplikacij
	*3-D kontrolnik*	Oblikovanje 3D učinkov na objektih

Tabela 7.2: Orodja v glavni orodni vrstici programa OpenOffice.org Draw

Prav tako lahko v risbo vstavimo rastrsko grafiko in sicer z ukazom "Vstavi→Datoteka" ali preko odložišča.

V glavni orodjarni so osnovna orodja za dodajanje in spreminjanje objektov na risbi. Predstavljena so v tabeli 7.2. Pod orodji se navadno skrivajo plavajoče orodjarne, ki se prikažejo po pritisku na določeno ikono, če nekaj časa držimo gumb miške pritisnjen. Če nameravamo posamezno plavajočo orodjarno večkrat uporabiti, jo lahko odnesemo na poljubno mesto na ekranu, kot je prikazano na sliki 7.10. Orodje v orodjarni ostane izbrano, dokler ga ne uporabimo. Po uporabi postane zopet aktivno prejšnje orodje. Če pa ga izberemo z dvojnim klikom, ostane orodje izbrano tudi po uporabi.

Posamezen objekt označimo z orodjem *Izberi*, ki se nahaja na vrhu glavne orodjarne. Če želimo označiti več objektov, med izbiranjem zadržimo tipko `<Shift>`. Objekte najhitreje brišemo tako, da jih označimo in pritisnemo tipko `<Delete>`.

Slika 7.10: Plavajoče orodjarne programa OpenOffice.org Draw

S predmetno orodjarno spreminjamo lastnosti obrobe in notranjosti ustvarjenega objekta. Vse lastnosti so zbrane v orodjih "Oblika→Črta" in "Oblika→Področje", nekatere od njih pa se nahajajo že v sami predmetni orodjarni.

Kot primer uporabe bomo narisali nekaj objektov, podobnim tistim na sliki 7.11.

Slika 7.11: Primer ustvarjanja objektov

- Izberimo orodje *Pravokotnik* iz glavne orodjarne in narišimo pravokotnik na risbo. Spremenimo njegovo obrobo v predmetni orodjarni, tako da bo ta črtkana, njena debelina "0,5 cm", barva pa "Črna". Podobno spremenimo barvo površine v "Svetlo sivo". V pravokotnik lahko z dvojnim klikom vpišemo besedilo.
- Podobno narišimo 3D objekta z orodjem *3-D Predmeti*. Izberimo orodje in odvlecimo plavajočo orodjarno na namizje. Narišimo objekta *Kolobar* in *Stožec*. Spremenimo barvo njune površine v "Rdeča" in "Svetlo rdeča". Tridimenzionalni objekt lahko tudi poljubno rotiramo v prostoru, če nanj kliknemo dvakrat zapored.
- Objekte premaknimo z miško v položaj, ki je prikazan na sliki.

Spreminjanje položaja, velikosti in rotacije

Položaj objekta spremenimo tako, da premaknemo (povlečemo) točko iz njegove notranjosti ali roba.

Če izberemo objekt z miško, se prikaže osem ročic za spreminjanje njegove velikosti. Ko miško premaknemo na eno od njih, se spremeni njena podoba. Takrat lahko z vlečenjem spremenimo velikost objekta.

Rotacijo spremenimo tako, da dvakrat kliknemo objekt (če izberemo ikono *Način vrtenja po kliku na predmet* iz orodne vrstice *Vrstica možnosti* na dnu zaslona, zadošča en klik na objekt). V sredini objekta se prikaže točka vrtenja, okoli katere lahko vrtimo objekt bodisi s pomočjo ročic v ogliščih izbranega objekta ali pa z vlečenjem poljubne točke v njem. Položaj točke vrtenja lahko spreminjamo z miško. Posebno vrsto rotacije predstavljajo ročice, ki so na sredini robov izbranega objekta. Z njimi zavrtimo objekt okoli nasproti ležečega roba. Tako na primer s sredinsko spodnjo ročico vrtimo objekt okoli zgornjega roba.

S ponovnim klikom na objekt ponovno prikažemo ročice za spreminjanje velikosti. Sedaj lahko z dvojnim klikom vnesemo besedilo, ki se bo prikazalo v objektu.

Ko premikamo grafične objekte, lahko ostanejo posamezni deli risbe zabrisani. Izris osvežimo s pritiskom na `<Ctrl><Shift><R>`.

Urejenost in poravnava objektov

Pod urejenostjo objektov razumemo vrstni red prekrivanja. Objekti, ki se nahajajo v ospredju, zakrivajo posamezne dele objektov v ozadju. Objekte premikamo v ospredje in ozadje s plavajočo orodjarno *Razporedi* iz glavne orodne vrstice.

Če želimo pravokotnik iz slike 7.11 premakniti v ospredje, ga najprej izberemo in nato kliknemo na ikono *Razporedi* iz glavne orodjarne. V ospredje ga postavimo tako, da dvakrat izberemo *Postavi naprej*. To orodje postavi objekt za nivo bližje opazovalcu. S *Premakni v ospredje* lahko postavimo izbrani objekt takoj v ospredje, ostale objekte pa s tem premaknemo za en nivo globje. Podobno ravnamo, ko želimo z ikonama *Pošlji nazaj* in *Pošlji v ozadje* objekt postaviti za ostale na sliki. Prestavimo sedaj pravokotnik nazaj v ozadje.

Poravnavo objektov uporabljamo, ko želimo položaj skupine objektov poravnati na določenem robu ali jih srediniti. Spreminjamo jo iz plavajoče orodjarne *Poravnava*.

Poravnajmo objekte na sliki 7.11 na njihov levi rob. Označimo najprej vse objekte z miško, tako da zadržimo tipko `<Shift>`. V plavajoči orodjarni *Poravnava* izberimo ikono *Levo*. Z ikonama *Na sredini* in *Desno* objekte sredinimo ali poravnamo na desni rob. Na podoben način lahko poravnavamo objekte po vertikali.

Urejenost in poravnavo lahko spreminjamo tudi iz kontekstnega menija.

Vnos besedila

V risbo lahko dodajamo tekstovne objekte – besedilo. Oblikujemo ga podobno, kot v ostalih modulih paketa OpenOffice.org, v risbah pa ga lahko tudi rotiramo. Posebna vrsta besedila so legende, ki imajo poseben okvir s puščico.

Odprimo plavajočo orodjarno *Besedilo* in izberimo ikono *Besedilo*. V risbi označimo področje, kamor želimo vpisati besedilo. Z vlečenjem ali s tipko `<Shift>` in smernimi tipkami lahko označimo poljubni del besedila in ga oblikujemo (na primer iz kontekstnega menija). Ko končamo z vnosom, kliknemo izven vnosnega območja. Besedilo lahko ponovno spremenimo z dvojnim klikom na področje, ki ga zavzema tekstni objekt. Lastnosti področja besedila (obrobo, notranjost, rotacijo) lahko spremenimo podobno kot pri ostalih grafičnih objektih.

Z ikono *Prilagodi besedilo okvirju* je delo podobno, le da se velikost vnešenega besedila samodejno prilagodi velikosti okvirja.

Legenda je tekstovno področje, ki s puščico kaže na poljubno mesto v risbi. Narišemo jo z *Oblački* iz plavajoče orodjarne *Besedilo*. Najprej narišemo puščico, nato pa še okvir, ki bo vseboval besedilo. Obliko in velikost puščice spremenimo v kontekstnem meniju "Črta" ali v predmetni orodni vrstici z ikono *Slog puščice*.

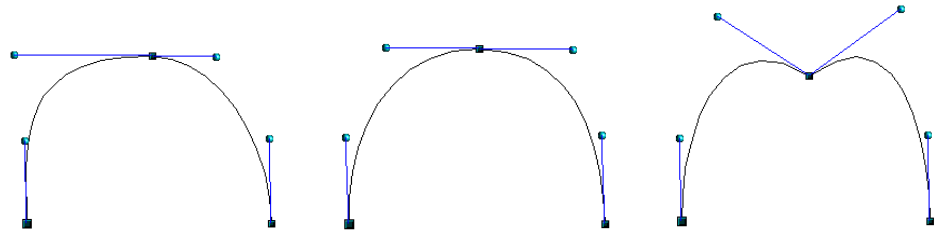
Besedilu lahko dodamo tudi 3D učinek. Izberimo orodje *Prilagodi besedilo okvirju* in označimo podolgovato pravokotno območje. Vanj vpišimo npr. besedilo `OpenOffice`. Iz kontekstnega menija izberimo "Pretvori→V 3D". Izgled učinka prikazuje slika 7.12.

Slika 7.12: Učinek "Pretvori v 3D"

7.2.2 Krivulje

Ena osnovnih možnosti vseh programov, ki delajo z vektorsko grafiko, je risanje krivulj poljubne oblike. OpenOffice.org Draw omogoča risanje t.i. Bézierjevih krivulj, ki so dobile ime po francoskem matematiku Pierru Bézierju. Vsako Bézierjevo krivuljo določajo točke, skozi katere ta poteka, in smerne ročice, ki opredeljujejo njeno ukrivljenost v posamezni točki, kot je prikazano na sliki 7.13. Dolžina smerne ročice določa prileganje krivulje njeni smeri. Daljšim ročicam se krivulja prilega bolj, krajšim pa

manj. Začetna in končna točka krivulje imata po eno smerno ročico, vse ostale pa dve.



Slika 7.13: Tri vrste nadzornih točk Bézierjeve krivulje

Ločimo tri vrste Bézierjevih točk:

1. Gladka točka je točka, v kateri je odvod zvezen, ukrivljenost pa ni nujno popolnoma enaka na obeh straneh. Smerni ročici gladke točke kažeta vedno v nasprotno smer, njuna dolžina pa je lahko različna. Primer gladke točke je sredinska točka leve krivulje na sliki 7.13.
2. Gladka simetrična točka je točka, v kateri je ukrivljenost enaka na obeh straneh. Smerni ročici gladke simetrične točke sta popolnoma enaki (kažeta v nasprotno smer in sta enako dolgi). Primer gladke simetrične točke je sredinska točka srednje krivulje na sliki 7.13.
3. V robni točki krivulja ni nujno gladka, ampak se lahko tudi prelomi. Smerni ročici robne točke sta lahko različni. Primer robne točke je sredinska točka desne krivulje na sliki 7.13.

Risanje krivulje

Orodja za risanje krivulj se nahajajo v plavajoči orodjarni *Krivulje*. Bézierjeve krivulje rišemo z orodjem *Krivulja*.

Slika 7.14: Risanje krivulje

Slika 7.14 prikazuje osnovne korake, ki so potrebni za risanje krivulje. Izberimo omenjeno orodje *Krivulja*. Mesto, kamor kliknemo najprej, določa položaj prve Bézierjeve točke krivulje. Nato z vlečenjem (gumba ne spustimo) narišemo smerno ročico začetne točke. Končno točko smerne ročice določa položaj miške, ko spustimo levi miškin gumb (glejte levi del

slike). Sedaj se lahko premaknemo z miško v položaj, kjer želimo drugo Bézierjevo točko. Opazimo lahko, da krivulja sledi miški, kot prikazuje srednji del slike. Drugo točko označimo podobno kot prvo. S klikom določimo položaj druge Bézierjeve točke, z vlečenjem pa usmerjenost in dolžino smerne ročice, kot je prikazano v desnem delu slike. Vse tako določene vmesne točke so gladke, začetni dolžini ročice pa sta enaki (izgledajo kot simetrične). Če želimo narisati robno točko, ne smemo vleči miške s pritisnjenim levim gubmom, ampak samo na hitro kliknemo. Če narišemo dve zaporedni robni točki s klikom, bo krivulja med njima ravna črta. Tako določena robna točka bo dejansko gladka, njeni smerni ročici pa imata dolžino nič. Risanje krivulje zaključimo z dvojnim klikom ali pa ga prekinemo s tipko `<Esc>`. Če zaključimo risanje v začetni točki, bo dobljena krivulja sklenjena.

Kadar med risanjem držimo tipko `<Shift>`, so smeri omejene na pravilne kote (mnogokratnike od 45°).

Najenostavnejšo vrsto krivulje narišemo z orodjem *Mnogokotnik*. Sestavljena je iz množice povezanih, ravnih črt. V tem primeru nadzorne točke nimajo smernih ročic. Krivulja, ki jo opiše poligon, ni gladka, temveč je lomljena.

Popravljanje krivulje

Izgled narisane krivulje lahko naknadno spremenimo in popravimo. Določamo lahko podobne lastnosti kot pri ostalih grafičnih objektih (debelino, barvo črte, oblikovanje notranjosti sklenjene krivulje, rotacijo itd). Vendar pa nas na tem mestu zanima predvsem popravljanje nadzornih točk krivulje.

Krivuljo najprej označimo. Nadzorne točke spreminjamo tako, da izberemo orodje *Uredi točke* na skrajni levi strani predmetne orodne vrstice na vrhu ali možnostne orodne vrstice na dnu zaslona. S tem aktiviramo Bézierjevo predmetno orodjarno, ki vsebuje dodatna orodja za spreminjanje nadzornih točk. Ko izberemo določeno nadzorno točko, se prikažeta njeni smerni daljici, na koncu katerih sta okrogli ročici. S premikanjem smernih ročic lahko poljubno spremenimo ukrivljenost v nadzorni točki. Ko izberemo nadzorno točko, se v Bézierjevi predmetni orodjarni prikaže, za kakšno vrsto nadzorne točke gre (robna, simetrična ali gladka). Vrsto nadzorne točke lahko poljubno spremenimo.

Z *Vstavi točke* in *Izbriši točke* dodajamo nove ali odvzamemo obstoječe nadzorne točke. Sklenjeno krivuljo razklenemo tako, da označimo točko, v kateri je krivulja zvezna, in izberemo *Razdeli krivuljo*.

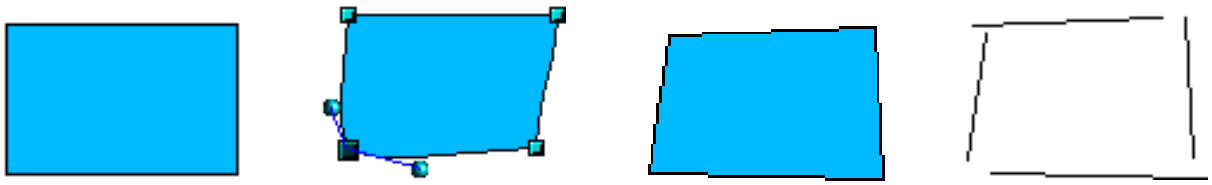
Kot je bilo že opisano ob sliki 7.13, ima vsaka Bézierjeva točka tri nadzorne točke. Točka, predstavljena z majhnim kvadratkom, določa njen položaj, okrogli točki pa smer in dolžino smernih ročic. Daljša kot je

smerna ročica, bolj se ji krivulja prilega.

Pretvarjanje grafičnih objektov v krivulje

Grafične objekte, ki jih rišemo z orodji iz glavne orodjarne (kvadrati, pravokotniki, krožnice, krogi, elipse, 3D objekti itd), lahko pretvorimo v Bézierjeve krivulje ali poligone. To nam omogoča, da lahko spremenimo katerokoli nadzorno točko, s čimer poljubno spremenimo obliko objekta.

Z orodjem *Pravokotnik* iz glavne orodjarne narišimo in izberimo pravokotnik. V Bézierjevo krivuljo ga pretvorimo z ukazom "Spremeni→Pretvori→V krivuljo" iz glavnega ali kontekstnega menija. Sedaj lahko lik poljubno spreminjamo, podobno kot spreminjamo Bézierjeve krivulje. Lik pretvorimo v poligon z ukazom "Spremeni→Pretvori→V mnogokotnik". Tudi poligon lahko poljubno popravimo. Povezan poligon lahko tudi razdremo v množico nepovezanih črt z ukazom "Spremeni→Prelomi". Vsi trije postopki so prikazani na sliki 7.15



Slika 7.15: Spreminjanje grafičnih objektov iz glavne orodjarne

Zanimiv učinek dobimo, če dvodimenzionalni objekt pretvorimo v 3D. Poizkusimo ustvariti izsek kroga z ukazom *Krožni tortni diagram*, ki se nahaja v plavajoči orodjarni *Elipse*. V kontekstnem meniju izberimo "Pretvori→V 3D". Ukaz doda drugo ploskev, ki je vzporedna prvotnemu liku. Objekt lahko sedaj rotiramo v treh dimenzijah.

7.2.3 Delo z objekti

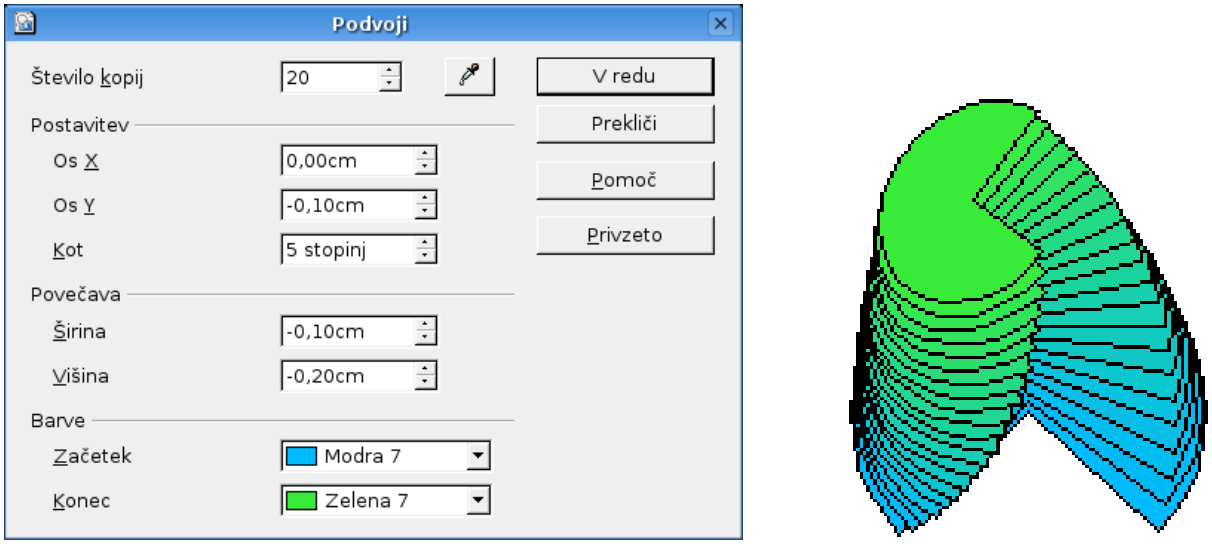
Grafični objekt najhitreje podvojite z odložiščem. Podobno kot pri delu s besedilom, lahko označite, prekopirate, izrežete in prilepite tudi grafični objekt. Vendar pa ima delo z grafičnimi objekti nekatere dodatne možnosti, ki pri besedilu niso na voljo. Te bodo opisane v nadaljevanju.

Podvajanje

Podvajanje objektov omogoča izdelavo množice objektov, pri čemer se vsaka kopija od prejšnje nekoliko razlikuje. Objekti se lahko razlikujejo po

položaju na risbi, velikosti, barvi ali orientaciji. Interval spremembe med dvema kopijama objekta je konstanten.

Kot primer bomo naredili podvajanje izseka iz elipse. Narišimo poljuben izsek iz elipse z orodjem *Krožni tortni diagram*. V glavnem meniju izberimo "Uredi→Podvoji". Prikaže se pogovorno okno *Podvoji*. V okvirju "Postavitev" prestavimo vsak naslednji objekt nekoliko višje, v "Povečava" ga nekoliko pomanjšajmo, spremenimo pa tudi barvo ciljnega objekta. Vsi popravki so prikazani na sliki 7.16. Ista slika prikazuje tudi rezultat, ki ga dobimo, ko izberemo "OK".

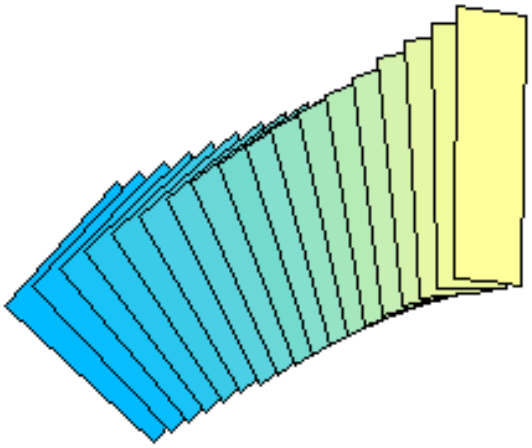


Slika 7.16: Podvajanje objekta

Prehajanje

Prehajanje (angl. *morphing*) je matematična operacija, ki izračuna množico prehodov med dvema objektoma. Objekti, ki so na risbi bližje prvemu, so mu po obliki, orientaciji, barvi in ostalih lastnostih bolj podobni.

Narišite dva različna pravokotnika. Drugemu spremenimo orientacijo in barvo, lahko pa tudi ostale lastnosti. Sedaj oba označimo in v glavnem meniju izberimo "Uredi→Prehajanje (ali Pojemanje)". V pogovornem oknu označite "Prehajanje atributov (ali Atributi pojemanja)" in izberite "OK". Pozor! V nekaterih različicah OpenOffice.org je prišlo do napake pri slovenjenju in napačni izrazi so navedeni v oklepajih. Slika 7.17 prikazuje primer prehajanja med dvema objektoma.



Slika 7.17: Prehajanje med dvema objektoma

Združevanje

Množico različnih objektov lahko izberemo ali združimo. Operacije nad izbranimi objekti so enakovredne operacijam na skupini istih objektov, dokler ne izberemo kakega drugega objekta. Takrat izbrana skupina razpade. Izbira množice objektov je torej začasna operacija. Če različne objekte združimo v skupine, ostanejo ti del skupine, dokler jih z novim ukazom zopet ne razbijemo na posamezne objekte. Vse operacije, ki jih izvajamo na tako ali drugače določeni skupini objektov, se izvedejo na vsakem posameznem objektu.

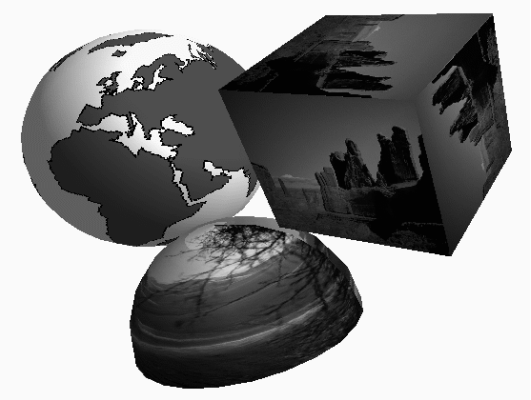
Primer združevanja bi lahko bila organizacijska shema podjetja. Najprej narišemo posamezne okvirje in vanje vpišemo besedilo. Nato okvirje povežemo v hierarhično organizacijsko shemo. Na tem mestu celotno risbo navadno združimo, ker želimo, da se vse operacije, ki jih bomo izvajali (na primer premik ali povečava), odražajo na vseh objektih v skupini.

Objekte najhitreje združimo tako, da jih izberemo in v kontekstnem meniju poiščemo ukaz “Združi”. Z ukazom “Razdruži” skupino ponovno razbijemo na posamezne objekte.

7.2.4 Spreminjanje barve, teksture

Barvo objekta ali njegove obrobe določamo v predmetni orodjarni, ki se nahaja nad risbo. Barve v padajočih menijih *Barva črte* in *Področni slog/polnilo* lahko prikažemo tudi v samostojni orodjarni v spodnjem delu zaslona. Orodjarno aktiviramo v glavnem meniju z “Pogled→Orodne vrstice→Barvna vrstica”.

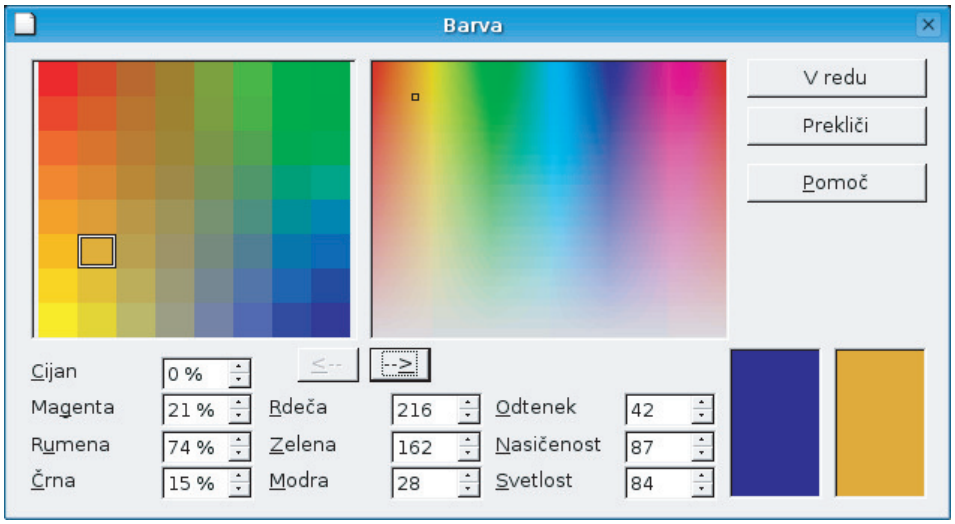
Barvna orodjarna prikazuje množico vnaprej pripravljenih barv, vanjo pa lahko dodajamo tudi svoje. Postopek bo opisan v naslednjem poglavju.



Slika 7.18: Lepljenje teksture na objekt

Zanimiva možnost je dodajanje teksture na objekt. Tekstura je bitna slika, v katero objekt "oblečemo". Če v predmetni orodjarni izberemo orodje *Področje*, v pogovornem oknu pa zavihek "Bitne slike", lahko z gubmom "Uvozi" izberemo sliko, ki jo želimo prilepiti na površino objekta. Ko potrdimo izbiro, se slika doda v seznam. Izberemo dodano sliko, se prestavimo na list "Področje" in izbrišemo izbiro "Razpostavi". Teksturo potrdimo z gubmom "OK". Nekaj primerov lepljenja je na sliki 7.18.

Dodajanje nove barve



Slika 7.19: Dodajanje nove barve v paleto

Če potrebujemo barvo, ki je ni v barvni paleti, jo lahko določimo sami na primer z ukazom "Oblika→Področje" iz glavnega menija. V pogovornem

oknu izberimo zavihek “Barve”, kot je prikazano na sliki 7.19.

Tabela v levem delu pogovornega okna prikazuje barve, ki se trenutno nahajajo v barvni paleti. Najprej izberimo barvo, ki je najbolj podobna želeni novi barvi. Prikazala se bo v vnosnem polju “Barva”. V “Ime” vpišimo ime nove barve. Barvni odtenek lahko spremenimo s spreminjanjem vrednosti “RGB” ali “CMYK”. Med obema barvnima formatoma izbirate v padajočem seznamu. Ker pa tovrstno spreminjanje ni najbolj udobno, si lahko pomagamo z gubmom “Uredi”. V novem pogovornem oknu *Barva* lahko z miško izberemo želeno barvo, vrednosti “RGB”, “CMYK” ali “HSV” pa se bodo nastavile samodejno. Izbiro barve potrdimo z “OK”, barvo pa v paleto vnesemo z gumbom “Dodaj” na zavihku “Barva”. Če pogovorno okno zapustimo s “Prekliči”, se barva trenutno izbranega objekta ne bo spremenila. Na novo dodano barvo lahko sedaj poiščemo v barvni paleti na dnu ekrana ali v padajočih menijih predmetne orodjarne.

7.2.5 Povezava programov GIMP in OpenOffice.org Draw

Za konec pa naredimo še en primer povezave obeh opisanih programov za delo z grafiko.

V programu GIMP ustvarimo novo sliko ločljivosti 600×600 in ji že pri ustvarjanju določimo prosojno ozadje. Z razpršilcem naredimo poljuben napis (na primer OpenOffice). Napis spremenimo tako, da bo izgledal, kot da smo ga naslikali s kredo. Sliko shranimo v formatu PNG. Sedaj ustvarimo novo risbo v programu OpenOffice.org Draw. Nanjo dodajmo 3D predmet Kolobar. Oblecimo ga v teksturo, ki smo jo ustvarili s programom GIMP.

Rezultat je na sliki 7.20.

7.3 GraphViz

V dokumentih često uporabljamo razne diagrame, ki temeljijo na grafih. Pod pojmom graf tu pojmujemo graf v smislu matematične teorije grafov. V teoriji grafov je graf G definiran z množico vozlišč V in množico povezav E med temi vozlišči:

$$\begin{aligned} G &= \{V, E\} \\ E &\subseteq V \times V \end{aligned}$$

Množici E včasih rečemo tudi *relacija* na V. Graf je usmerjen, če nas zanima smer povezav v E, in neusmerjen, če nas smer ne zanima.

Slika 7.20: Primer povezave programov GIMP in OpenOffice.org Draw

Vozliščem in povezavam lahko pripišemo dodatne atribute, npr. ceno, barvo, obliko, napis...

Graf lahko očitno narišemo kot diagram, pri čemer vozlišča narišemo s točkami ali liki, povezave pa črtami ali puščicami. Poseben problem je naloga "kako čim lepše narisati graf" iz dane množice vozlišč in povezav.

Pri avtomatičnem risanju grafov si lahko pomagamo s paketom programov GraphViz, ki so jih napisali v Bellovih laboratorijih, kjer so razvili tudi prvi sistem Unix. Telekomunikacijska podjetja pogosto rišejo diagrame kompeksnih sistemov in omrežij.

Paket vsebuje prevajalnika `dot` in `neato`, ki prevedeta opis grafa v tekstovni datoteki v slikovno datoteko, navadno v zapis PostScript ali SVG. Program `dot` skuša graf narisati po vrsti, kakor je definiran v datoteki, `neato` pa uporablja fizikalni model uteži povezanih z vzmeti, nato pa z optimizacijo skuša doseči čim bolj enakomerno razporeditev. Navadno se `dot` bolje obnese za manjše, `neato` pa za nekoliko večje grafe.

Primer: V datoteko `vaja.dot` shranimo naslednje besedilo:

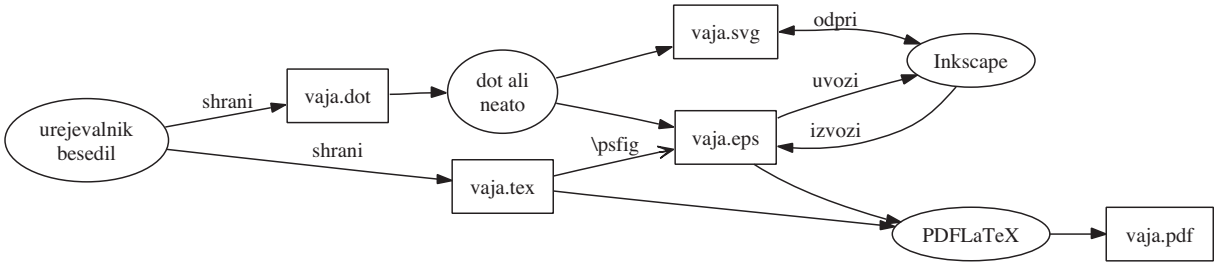
```
digraph i {
/* atributi celotnega grafa */
 size="5,5"; rankdir=LR; overlap=false;
/* vozlisca */
 graphviz [label="dot ali\nneato"];
 editor [label="urejevalnik\nbesedil"];
 draw [label="Inkscape"];
 LaTeX [label="PDFLaTeX"];
```

```
 node[shape=rectangle]; /* privzeti atributi za nova vozlisca */
/* povezave */
 editor -> "vaja.dot" [label="shrani"];
 editor -> "vaja.tex" [label="shrani"];
 "vaja.dot" -> graphviz;
 graphviz -> "vaja.eps"; graphviz -> "vaja.svg";
 "vaja.tex" -> LaTeX;
 "vaja.eps" -> LaTeX;
  LaTeX -> "vaja.pdf";
 "vaja.tex" -> "vaja.eps" [label="\\psfig",arrowhead=vee];
 "vaja.eps" -> draw [label="uvozi"];
 "vaja.svg" -> draw [label="odpri",arrowtail=normal];
  draw -> "vaja.eps" [label="izvozi"];
}
```

Spisek vseh podprtih atributov, podrobno dokumentacijo in primere lahko najdemo v priročniku (`man`) za `dot` ali `neato` oziroma na domači strani projekta (gl. razd. 7.5).

Datoteko prevedemo z ukazom `dot` ali `neato`. S stikalom `-T` določimo format izhoda, z `-o` pa izhodno datoteko:

```
  dot -Tps vaja.dot -o vaja_dot.eps
neato -Tps vaja.dot -o vaja_neato.eps
```

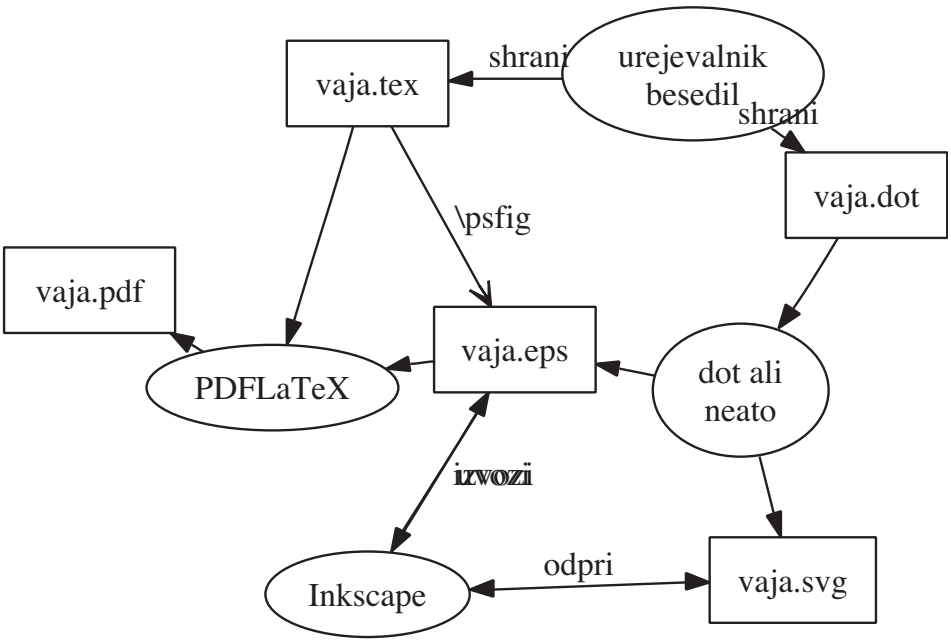


Slika 7.21: Graf postavljen z `dot`

Datoteke `.dot` zna tvoriti tudi vrsta drugih orodij za Unix. Za okolje X11 obstajata tudi inačici programov `dot` in `neato` s preprostim grafičnim vmesnikom - uporabiti moramo ukaza `dotty` in `lneato`. Za sistem MacOS X pa obstaja celo precej udoben in kakovosten grafični vmesnik, ki pomaga pri urejanju atributov grafa in ima aktivni predogled.

7.4 Naloge

1. Po kakšnih kriterijih ločujemo grafične računalniške programe?



Slika 7.22: Graf postavljen z `neato`

2. Kakšna je razlika med rastrsko in vektorsko računalniško grafiko? Kako so pri vsaki zapisani objekti?

3. Pojasni pojma uporabniški grafični program in grafična podatkovna struktura.

4. Kaj je PostScript?

5. Kakšna je namembnost uporabniških grafičnih programov? Naštej nekaj znanih programov in njihovo namembnost.

6. Kako je v programu GIMP sestavljen slikovni dokument? Kako lahko spreminjamo velikost in globino barv? Kako varčujemo pri velikosti slike?

7. Preizkusite risalna orodja in poskušajte čimbolj verodostojno napisati svoj podpis! Katerih lastnosti pisave z miško ne moremo verodostojno oponašati?

8. Stare slike so sivinske, zaradi staranja pa so porumenele. Vzemite kakšno barvno sliko in jo obdelajte tako, da bo izgledala starinsko.

9. S kakšnimi pripomočki si pomagamo pri oblikovanju krivulj? Poskusite narisati črko B z Bézierjevo krivuljo! Najmanj koliko nadzornih točk potrebujete?

10. Preizkusite operacijo podvajanja na 3D besedilu!

11. Poskusite operacijo prehajanja med različnimi mnogokotniki!

12. Poskušajte narisati tloris računalniške učilnice!

13. S paketom GraphViz narišite družinsko drevo, ki vsebuje vaše bližnje sorodnike! Puščice naj označujejo sorodstvena razmerja.

7.5 Koristne spletne povezave

1. Spletna stran programa GIMP:
   `http://www.gimp.org/`

2. Spletna stran OpenOffice.org:
   `http://www.openoffice.org/`

3. Spletna stran slovenskega OpenOffice.org:
   `http://sl.openoffice.org/`
   (S te strani je pod licenco GNU/FDL dostopna tudi knjiga [4].)

4. PostScript:
   `http://www.adobe.com/products/postscript/main.html`

5. GraphViz @ AT&T:
   `http://www.research.att.com/sw/tools/graphviz/`

6. Projekt GraphViz:
   `http://www.graphviz.org/`

7.6 Literatura

[1] Adobe Systems. *PostScript Language Tutorial and Cookbook*. Addison-Wesley, 1985.

[2] J. Foley, A. van Dam, S. Feiner, J. Hughes, and R. Phillips. *Introduction to Computer Graphics*. Addison-Wesley, Reading, MA, 1993.

[3] S. Haugland and F. Jones. *OpenOffice.org 1.0 Resource Kit*. Prentice-Hall PTR, Upper Saddle River, NJ, 2003.

[4] G. Leete, E. Finkelstein, and M. Leete. *OpenOffice.org for Dummies*. Wiley Publishing, Inc., Hoboken, NJ, 2003.

[5] R. Ludvik, I. Zajc, and A. Medic. *Hitri vodnik po OpenOffice.org*. Pasadena, Ljubljana, 2003. (Dostopna pod licenco GNU/FDL na: http://openoffice.lugos.si/knjiga/Hitri_vodnik_po_OpenOffice.org_FDL.pdf)

[6] R. C. Parker. *Grafično oblikovanje.* Pasadena, Ljubljana, 1997.

[7] E. R. Tufte. *The Visual Display of Quantitative Information.* Graphics Press, Cheshire, CT, 1983.

[8] E. R. Tufte. *Envisioning Information.* Graphics Press, Cheshire, CT, 1990.

[9] E. R. Tufte. *Visual Explanations.* Graphics Press, Cheshire, CT, 1997.

8

Govorne predstavitve

V poslovnem in akademskem svetu moramo pogosto predstaviti rezultate svojega dela ali naše poglede na določen problem pred ožjim ali širšim občinstvom. Da bo naša predstavitev uspešna, se moramo nanjo dobro pripraviti. Upoštevati moramo predvsem, kdo nas bo poslušal in koliko časa imamo na voljo za predstavitev. Čas za predstavitev je običajno vnaprej strogo omejen, še posebej, če nastopamo v okviru konference, kjer se morajo vsi udeleženci strogo držati urnika. Za svoj nastop moramo pripraviti ustrezna gradiva in svoj nastop vaditi. Najboljša formula za uspešen nastop je, da smo jasni in preprosti, ter da se predstavimo takšni, kot v resnici smo.

Gradiva, ki jih moramo pripraviti za govorno predstavitev, delimo na:

Gradiva za udeležence. To so gradiva, ki jih razdelimo udeležencem že pred ali med samim nastopom. Poleg vabila pošljemo običajno tudi kratke teze ali povzetek našega predavanja. Včasih pa dodamo tudi kakšno bolj izčrpno gradivo, kot so članki in poročila. Obsežnejša gradiva ali testne programe lahko razdelimo na zgoščenkah ali objavimo gradivo na internetu. Na strokovnih in znanstvenih konferencah običajno organizator poskrbi za konferenčni zbornik, v katerem so zbrani pisni prispevki vseh predavanj, in ga razdeli vsem udeležencem.

Gradiva za govornika so namenjena nastopajočemu bodisi pri pripravi na nastop ali za sam nastop. Pripravimo si lahko besedilo referata za branje, ki ga lahko še dodatno opremimo z navodili za podajanje (skript). Ali je med dejanskim nastopom dopustno branje referata ali ne, je odvisno od vrste dogodka. Nastop je prav gotovo bolj spontan in neposreden, če prosto govorimo in ne beremo. Za pravilni vrstni red podajanja in kot pomoč pri našem podajanju pa si lahko pomagamo s projekcijskimi gradivi.

Projekcijska gradiva. Klasični medij za podporo predavanju je tabla, na katero pišemo bodisi s kredo ali drugimi vrstami svinčnikov. Zelo je razširjena tudi uporaba projekcije s pomočjo grafoskopa, s katerim projeciramo folije oziroma prosojnice, ali pa diaprojektorja, s katerim projeciramo diapozitive. Prosojnice ali diapozitive lahko pripravimo s pomočjo urejevalnikov besedil in drugih grafičnih programov na računalniku. V zadnjem času pa je vse dostopnejša in vse bolj razširjena projekcija neposredno z računalnika s pomočjo video projektorja.

Na tem mestu bomo podali le nekaj osnovnih navodil za pripravo projekcijskih gradiv. Glede na razpoložljivi čas moramo najprej narediti izbor tega, kar želimo povedati. Ker temelji večina takih govornih predstavitev na osnovi nekega daljšega pisnega dokumenta ali nekega obsežnejšega produkta, moramo skrbno pretehtati, kaj lahko oziroma moramo povedati v omejenem času, kaj pa moramo izpustiti.

Preprosto pravilo pravi, da za eno minuto govora pripravimo eno prosojnico. Na eno prosojnico pa lahko napišemo le toliko, kolikor je možno napisati s tiskanimi črkami na listek papirja velikosti 8×8 cm. Zaželeno je, če lahko čim več informacij predstavimo na grafičen način.

Če uporabljamo prosojnice na plastičnih folijah, za besedilo uporabimo temno pisavo na svetlem ozadju. Če pa uporabljamo diapozitive ali projekcijo z računalnika, potem lahko uporabimo tudi svetlo pisavo na temnem ozadju. Črke morajo biti dovolj velike in na primernem ozadju, tako da so lahko čitljive.

Če uporabljamo diapozitive ali računalniško projekcijo s svetlo pisavo na temnem ozadju in različne barve, mora biti prostor bolj zatemnjen, da poslušalci lahko dobro razločijo projecirane informacije. Zatemnjen prostor pa lahko postane pri dolgih predstavitvah utrujajoč. Pri pripravi računalniške projekcije moramo paziti na pravo mero, saj tehnologija in različni učinki lahko kaj hitro zasenčijo govornika, ki pa bi moral vsemu navkljub ostati v središču dogajanja. Pri uporabi zahtevne tehnologije moramo biti pripravljeni tudi na morebitne tehnične težave. Zato je koristno za rezervo prosojnice tudi natisniti na folije, ki jih lahko pokažemo na navadnem grafoskopu.

Računalniško projekcijo lahko pripravimo na več načinov. Uporabimo lahko posebne programe za izdelavo multimedijskih predstavitev, kot je OpenOffice.org Impress, ki ga bomo opisali v nadaljevanju, ali pa izdelamo kar spletno predstavitev v jeziku HTML, ki jo tudi lahko opremimo s slikovnim materialom in videom. Z vse večjo zmogljivostjo spletnih brskalnikov, lahko tudi spletno predstavitev opremimo z interaktivnostjo in posebnimi učinki.

Lahko pa se zgodi, da zaradi nezdružljivosti spletnih brskalnikov naša

predstavitev na drugem računalniku ne bo izgledala tako, kot smo si jo zamislili. Da bi dosegli kar največjo združljivost našega dokumenta, lahko prestavitev izdelamo v formatu Adobe PDF (Portable Document Format). Format PDF lahko beremo z brezplačnim programom Acrobat Reader, ki na vseh platformah prikaže dokument točno tako, kot je bil zamišljen.

Predstavitev v obliki PDF lahko izdelamo s pomočjo sistema LaTeX. Potrebujemo le dodatek pdfLaTeX, ki dokument LaTeX pretvori v PDF, ter stilno datoteko `pdfslide.sty`, v kateri so opisani tudi dodatni ukazi, ki omogočajo izdelavo interaktivne predstavitve.

V nadaljevanju bomo opisali program OpenOffice.org Impress, s katerim lahko računalniške predstavitve izdelamo na enostaven in hiter način, ki je zelo podoben vizualnemu urejanju besedil.

8.1 OpenOffice.org Impress

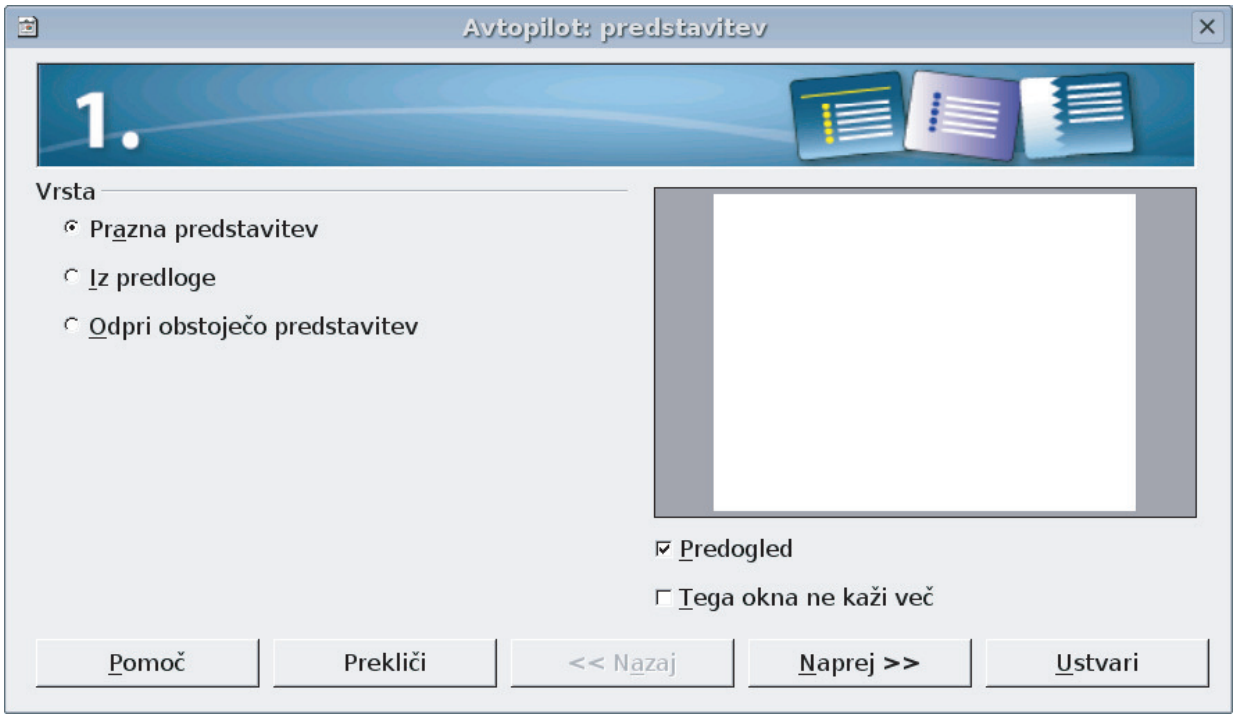
OpenOffice.org Impress je modul za izdelavo predstavitev v pisarniškem paketu OpenOffice.org. Z njim lahko izdelamo sosledje predstavitvenih strani, ki jih uporabimo za računalniško projekcijo, ali pa jih natisnemo in uporabimo kot prosojnice na navadnem grafoskopu. Predstavitev lahko tudi izvozimo v dokument HTML in jo objavimo na spletu.

V predstavitev vključujemo besedilo, grafiko, preglednice, baze podatkov, zvočne učinke, animacije, povezave na spletne strani in druge elemente. Predstavitve z računalniško projekcijo so privlačne predvsem zaradi multimedijskih učinkov in interaktivnosti, ki jih ostali načini predstavitev ne omogočajo. Ker so podrobnosti o delu z besedilom, vektorsko in rastrsko grafiko ter preglednicami in tabelami opisane v ostalih poglavjih knjige, bomo v opisu programa OpenOffice.org Impress poudarili prav vključevanje multimedije in interaktivnosti.

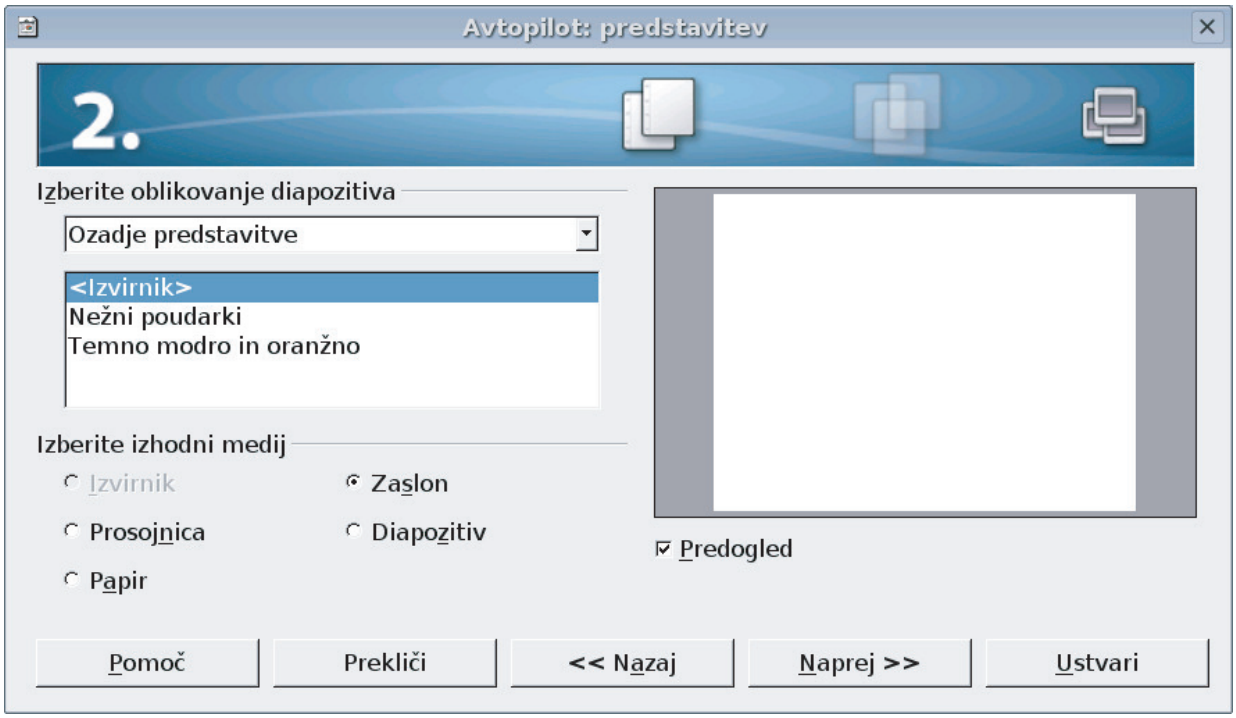
8.1.1 Začetek dela z orodjem OpenOffice.org Impress

Ko zaženemo Impress ali izberemo “Datoteka→Nov→Presentation” se nam prikaže okno čarovnika, ki se imenuje Avtopilot (slika 8.1). To okno lahko tudi izklopimo, tako da v prvem pogovornem oknu izberemo opcijo: “Tega okna ne kaži več”. Tako bomo naslednjič začeli s popolnoma prazno predstavijo. Okno čarovnika pa bomo lahko priklicali z izbiro “Datoteka→Avtopilot→Presentation”

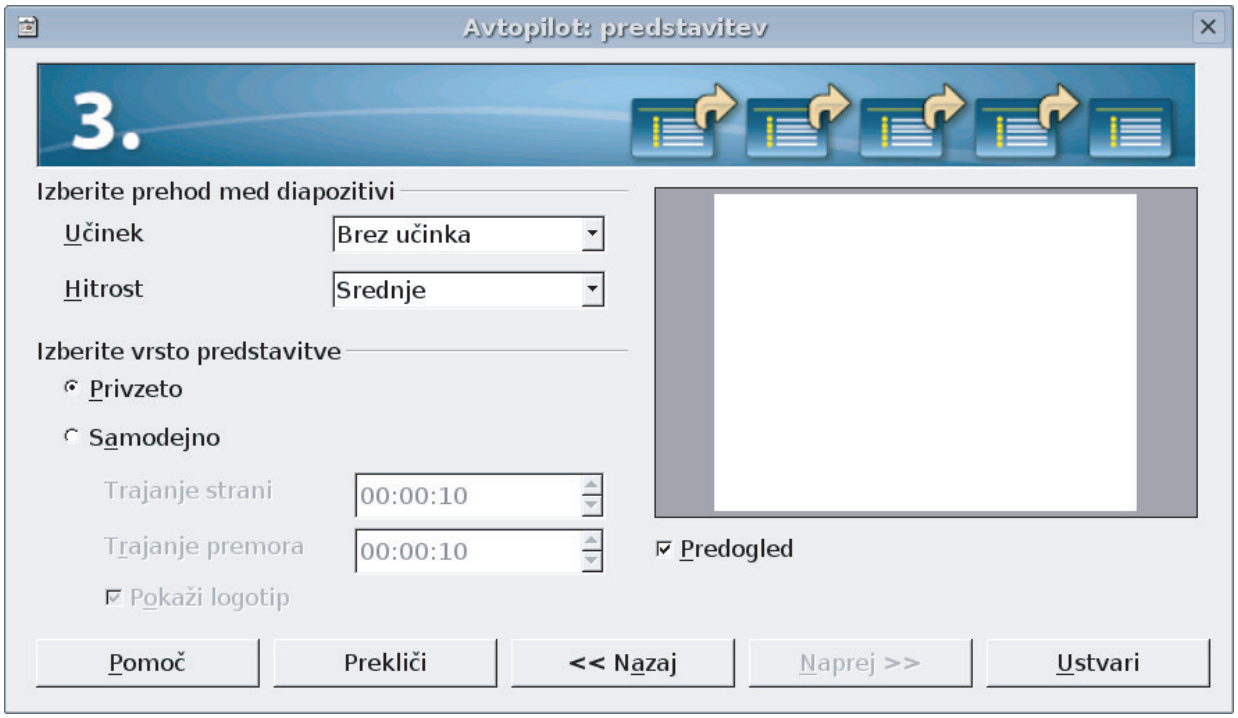
V prvem oknu čarovnika določimo, ali želimo začeti s praznim dokumentom, iz predloge ali pa želimo odpreti shranjeno predstavitev. Če izberemo predlogo, moramo izbrati tudi tip predstavitve. V drugem oknu (slika 8.2) izberemo izgled predstavitve, ali, če nočemo posebnega izgleda, izberemo “Izvirnik”. Za izhodni medij pa izberemo, kako bomo



Slika 8.1: Prvo pogovorno okno čarovnika za izbiro vrste predstavitve



Slika 8.2: Drugo pogovorno okno čarovnika za izbor izgleda predstavitve



Slika 8.3: Nastavitev prehodov med stranmi in časovnih parametrov

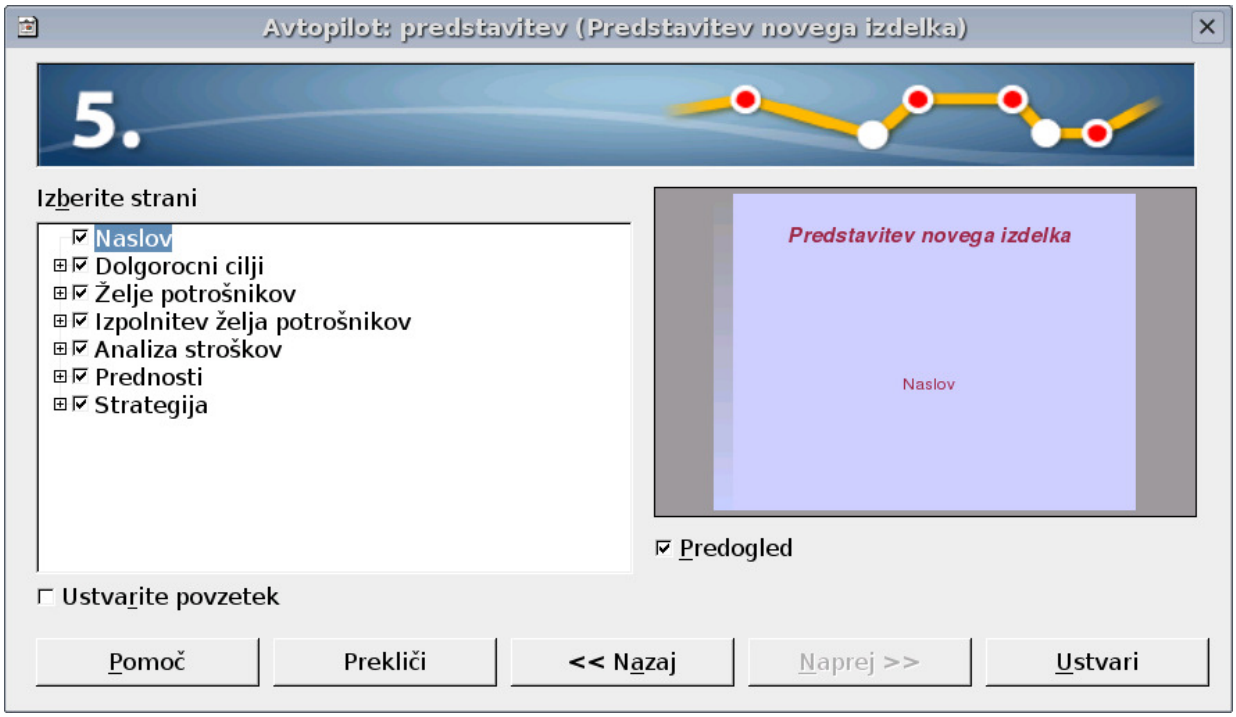
predstavitev predstavili. Običajno je to na računalniškemu zaslonu.

V tretjem pogovornem oknu (slika 8.3) določimo prehode med stranmi, ter – če želimo, da bo predstavitev samodejna – intervale prehajanja v sekundah. Četrto in peto pogovorno okno se nam prikaže samo, če smo izbrali na začetku predlogo predstavitve. V četrtem oknu opišemo osnovno idejo naše predstavitve, v petem oknu (slika 8.4) pa vidimo hierarhično strukturo predloge izbrane predstavitve; tu si jo lahko ogledamo ter izločimo nepotrebne strani. S klikom na gumb “Ustvar”i začnemo z urejanjem.

Če smo delo začeli iz predloge, se v oknu prikaže zadnja stran predstavitve. Ker smo v predlogi dobili tudi vsebinsko osnovo predstavitve, je že na začetku sestavljena iz več strani. Po straneh se premikamo s tipkama `<Page Up>` in `<Page Down>`.

Če nam predlagana predstavitev ustreza tudi vsebinsko, nam ne preostane drugega, kot da privzeto besedilo zamenjamo z našim besedilom, in preprosta predstavitev je že pripravljena.

Če pa hočemo vsebinski koncept predstavitve oblikovati sami, v prvem pogovornem oknu čarovnika (slika 8.1) izberemo, da želimo začeti s prazno predstavitvijo. V tem primeru moramo, preden se prikaže prva stran, izbrati tudi njen izgled (slika 8.5).



Slika 8.4: Pregled strukture predstavitve

Stran bo vsebovala okvirje, v katere lahko vnašamo besedilo in druge vsebine. Ob strani se aktivira glavna orodna vrstica modula OpenOffice.org Impress (tabela 8.1).

8.1.2 Osnove dela s predstavitvijo

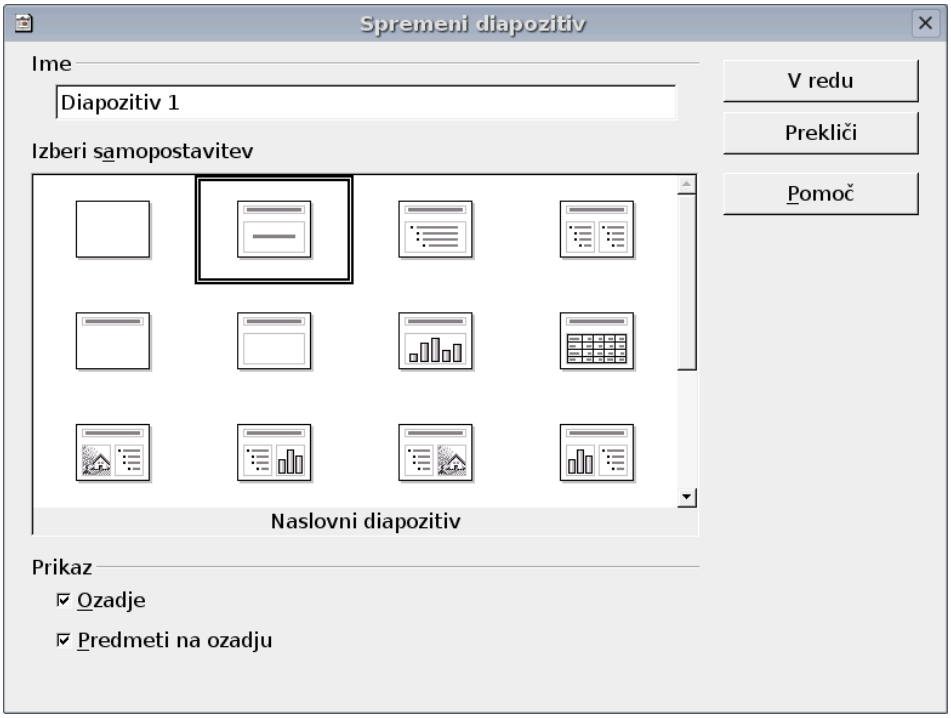
Če delo začnemo s prazno predstavitvijo, pa tudi sicer, moramo v predstavitev vstaviti nove strani. Nekatere strani želimo kasneje tudi odstraniti.

Novo stran vstavimo tako, da iz kontekstnega menija trenutne strani izberemo "Diapozitiv→Vstavi diapozitiv" ali pa kar z izbiro "Vstavi→Diapozitiv" iz glavnega menija. Najenostavneje pa lahko vstavimo novo stran s pomočjo predstavitvenega okna, da izberemo opcijo "Vstavi diapozitiv". Če predstavitvenega okna nimamo vključenega, ga lahko vključimo z izbiro "Pogled→Orodne vrstice→Predstavitev". V vseh primerih se prikaže še pogovorno okno (slika 8.5), v katerem izberemo okviren izgled strani. Izbiramo lahko med 20 predlogami, vsaka pa predstavlja drugačno razporeditev naslova, besedilnih in predmetnih okvirjev.

Trenutno stran odstranimo iz predstavitve na podoben način. V kontekstnem meniju izberemo "Diapozitiv→Izbriši diapozitiv" ali pa v

Ikona	*Ime orodja*	*Opis*
	`Izberi`	Označevanje objekta ali skupine objektov
	`Zoom`	Povečava in pomanjšava pogleda
	`Besedilo`	Oblikovanje besedila v risbi
	`Pravokotnik`	Risanje pravokotnikov
	`Elipsa`	Risanje elips, krogov, krožnih izsekov
	`3D predmeti`	Risanje 3D objektov
	`Krivulja`	Risanje krivulj, poligonov
	`Črte in puščice`	Risanje črt in puščic
	`Konektor`	Risanje povezav
	`Zasukaj`	Zasuk objektov
	`Poravnava`	Poravnava objektov
	`Razporedi`	Razvrščanje objektov
	`Vstavi`	Vstavljanje objektov iz drugih aplikacij
	`Animacijski učinki`	Posebni učinki
	`Interakcija`	Interaktivnost
	`3D kontrolnik`	Oblikovanje 3D učinkov na objektih
	`Diaprojekcija`	Zagon predstavitve

Tabela 8.1: Orodja v glavni orodni vrstici programa OpenOffice.org Impress



Slika 8.5: Pogovorno okno za izbor izgleda strani

glavnem meniju “Uredi→Izbriši diapozitiv”.

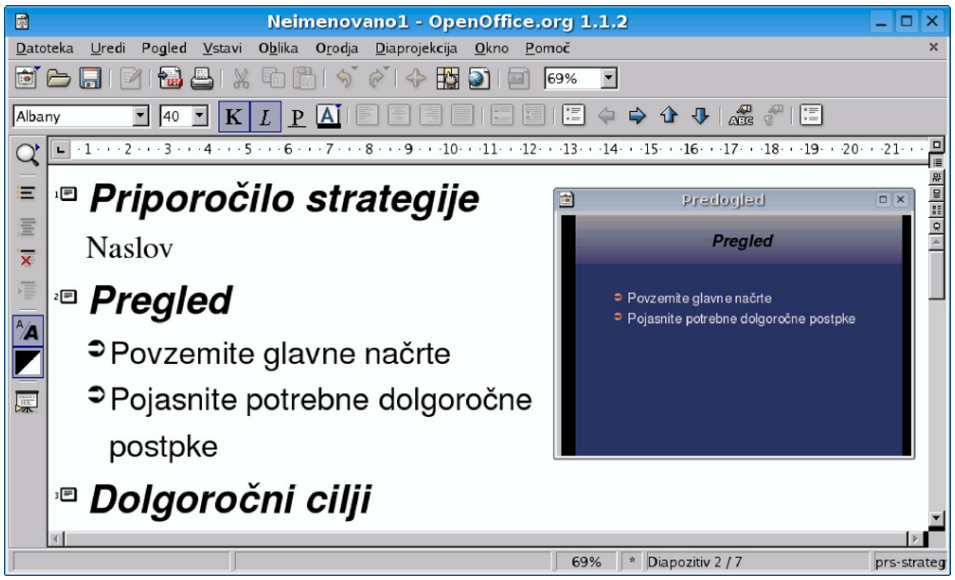
Načini urejanja predstavitve

Med urejanjem predstavitve si pomagamo s preklapljanjem med različnimi načini prikaza na zaslonu. Med načini preklapljamo z gumbi, ki se nahajajo ob desnem robu okna (tabela 8.2).

Ikona	*Način*	*Opis*
	`Risalni pogled`	Osnovni prikaz strani
	`Orisni pogled`	Hierarhični način z besedilom
	`Pogled diapozitiva`	Razvrščevalnik strani
	`Pogled opomb`	Omogoča vnos opomb k stranem
	`Pogled izročka`	Tu specificiramo način izpisa

Tabela 8.2: Načini urejanja predstavitve

Besedila v dokumentu najlaže urejamo v orisnem pogledu (slika 8.6), še



Slika 8.6: Orisni pregled predstavitve

posebej naslove. Tu lahko dele besedila premikamo po dokumentu, jim določimo globino v strukturah (s puščicami v predmetni orodni vrstici), lahko pa spreminjamo tudi vrstni red strani. Vsakič, ko pritisnemo tipko `<Enter>`, se pojavi kazalec za vnos na istem nivoju (angl. *outline*) kot v prejšni vrstici, npr. na nivoju glavnega naslova. Če želimo pod prejšnji naslov vnesti še podrejeno besedilo, pritisnemo tipko `<Tab>` in vrstico bomo degradirali v podnaslov. Vrstico pa lahko promoviramo s kombinacijo `<Shift><Tab>`. Tako lahko na primer iz podnaslova `Podnaslov`, ki je hierarhično na tretjem nivoju, dobimo novo stran (prvi nivo) z naslovom `Podnaslov` le z dvema pritiskoma na `<Shift><Tab>`.

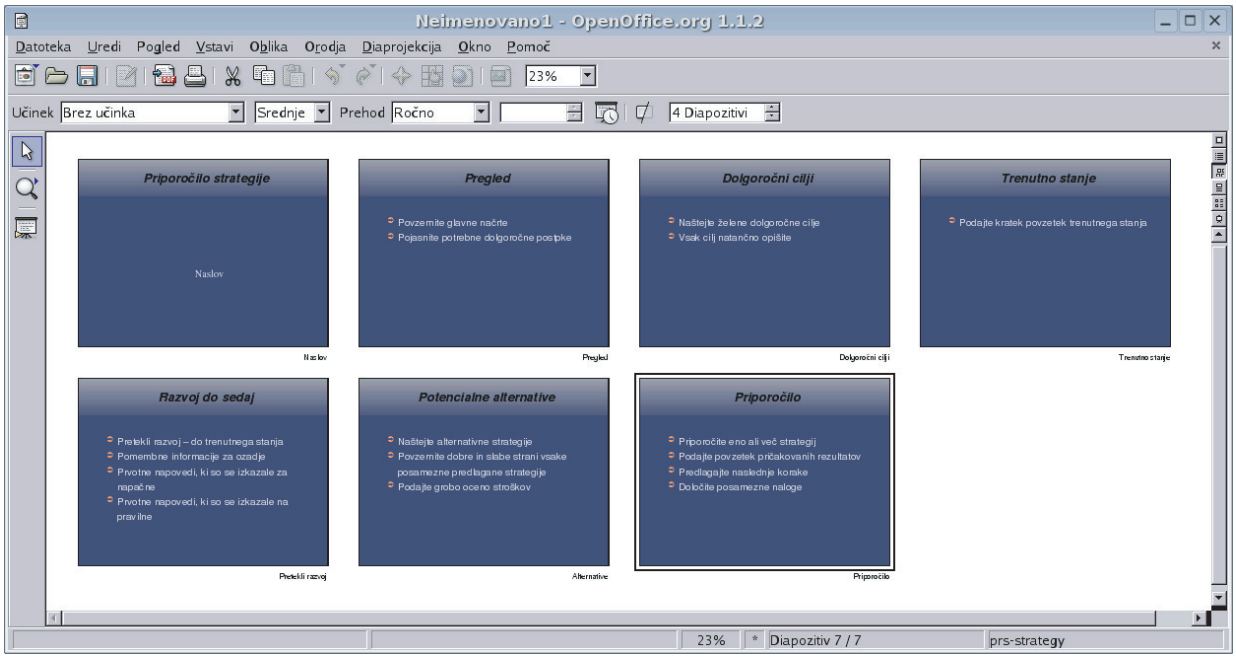
Vrstni red strani pa najlaže določamo v razvrščevalniku strani (*Pogledu diapozitiva*, slika 8.7). V razvrščevalniku imamo dober pregled nad sosledjem strani, zaporedje pa spreminjamo z akcijo vleci in spusti. Strani lahko tu tudi brišemo z uporabo kontekstnega menija.

Posamezno stran lahko tudi le začasno "skrijemo" z ukazom "Diaprojekcija→Prikaži/skrij diapozitiv" ali pa s "Prikaži/skrij diapozitiv" iz kontekstnega menija. Ime skrite strani bo v razvrščevalniku izpisano sivo; ob prikazovanju predstavitve bo ta stran izpuščena.

8.1.3 Slogi in ozadja

Računalniške predstavitve navadno obogatimo z grafičnimi dodatki, ki se skozi predstavitev ne spreminjajo, temveč ostajajo v ozadju.

Ozadje lahko definiramo na več načinov. Opisali bomo t.i. *Matrični*



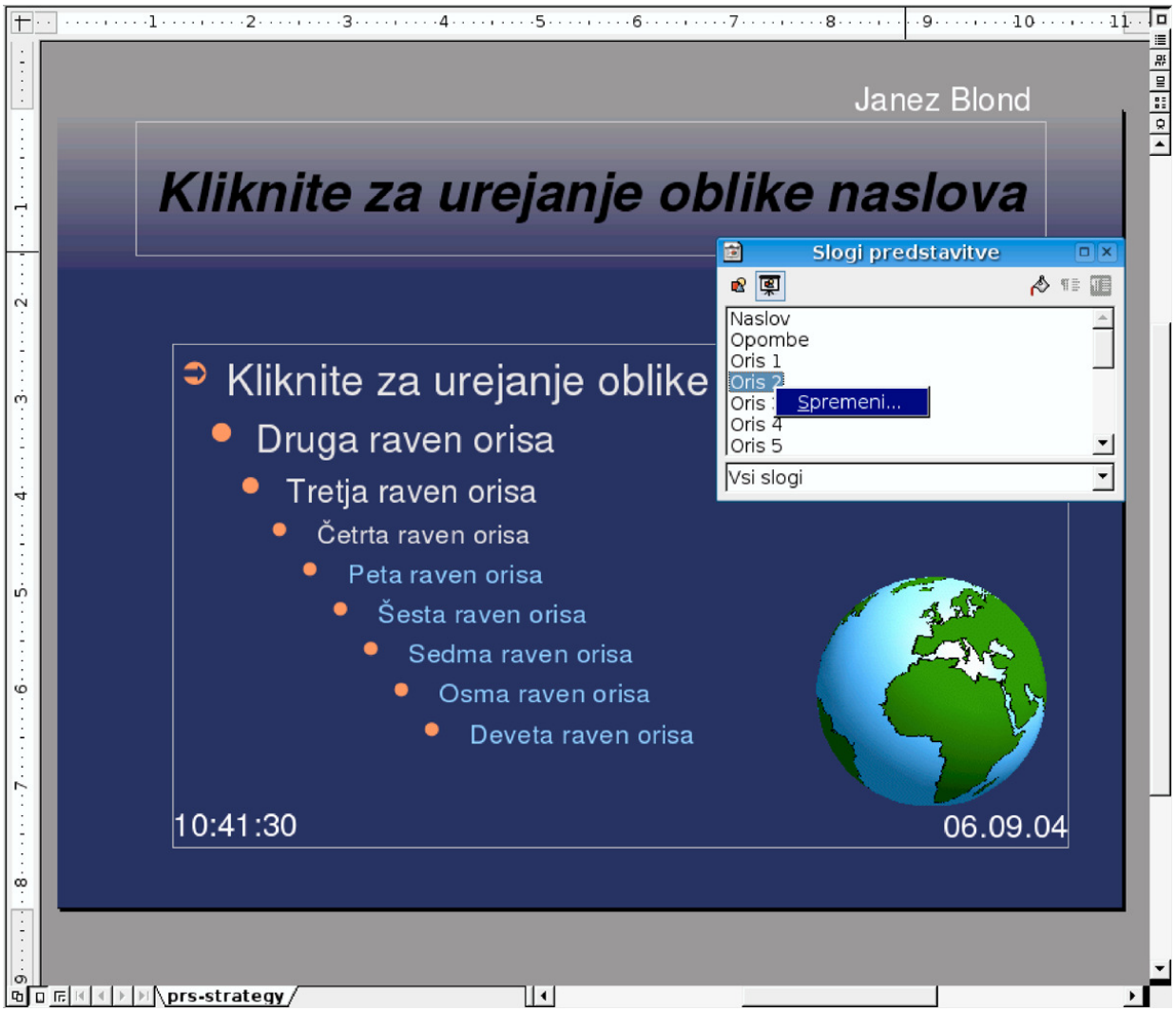
Slika 8.7: Razvrščevalnik strani

pogled, v katerem lahko spreminjamo tudi izgled celotne predstavitve.

V *Matrični pogled* preklopimo s klikom na ikono v spodnjem levem kotu okna. Prikaže se generična stran, ki vsebuje ozadje ter besedilo, ki predstavlja različne sloge za celotno hierarhijo besedila. Ozadje spreminjamo tako, da dodajamo grafične elemente, slike, polja in besedilo.

Na sliki 8.8 vidimo, da smo iz galerije "Orodja→Galerija" izvirnemu ozadju dodali 3D lik Zemlje v spodnji desni kot, tja smo postavili tudi polje ("Vstavi→Polja") s trenutnim datumom, v zgornji desni kot smo postavili ime avtorja, v spodnji levi kot pa trenutni čas.

Na sliki vidimo tudi besedilo, ki služi kot vzorec za spreminjanje generičnega sloga strani. Glavni elementi strani so naslov, podnaslov, ozadje, predmeti ozadja, opombe in devet stopenj hierarhičnega besedila oziroma ravni orisov. Vsi ti elementi so dosegljivi v *Matričnem pogledu*; če na primer vrstici `Druga raven orisa` spremenimo lastnosti, s tem spremenimo generično stran in sprememba se bo poznala v celi predstavitvi. Na sliki vidimo, da si pri tem lahko pomagamo tudi s slogovnikom ("Oblika→Slogovnik"). Ustrezen element izberemo ter v njegovem kontekstnem meniju izberemo "Spremeni". Tako lahko denimo elementu "Oris 1" spremenimo velikost ali vrsto pisave, ozadje pa obarvamo ali zapolnimo z vzorcem.



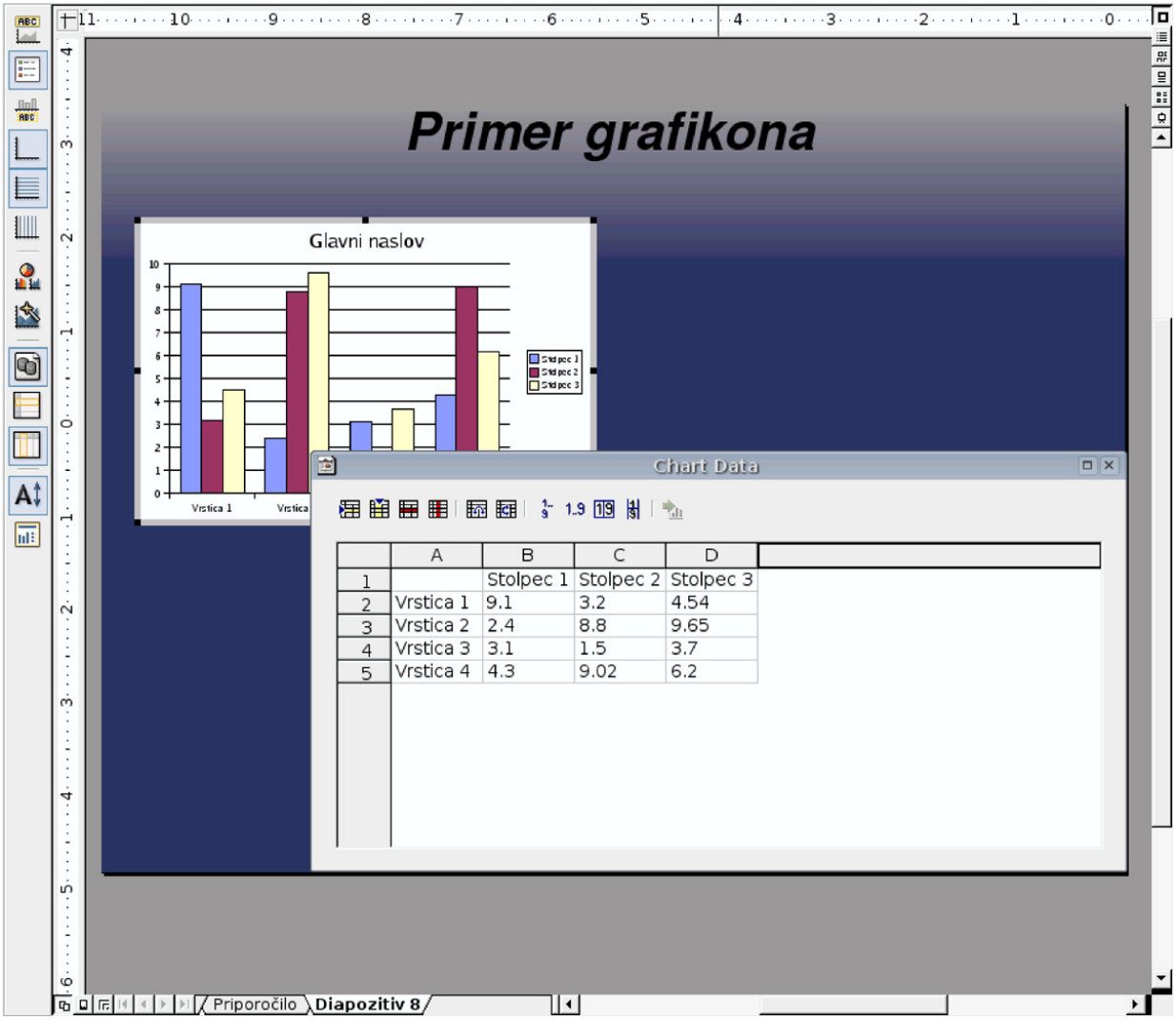
Slika 8.8: Spreminjanje generične strani

8.1.4 Vstavljanje drugih vsebin

Predstavitev lahko obogatimo z dokumenti iz drugih modulov, poleg tega pa jo lahko tudi poživimo z multimedijskimi vložki in interaktivnostjo.

Predmet OLE najenostavneje dodamo tako, da že pri vstavljanju nove strani v pogovornem oknu izberemo ustrezno obliko strani, ki vključuje tudi predmete. Ko se nova stran prikaže, kliknemo okvir, namenjen za dodajanje predmeta. Pojavi se pogovorno okno “Vstavi OLE predmet”. Predmet lahko bodisi ustvarimo sami ali pa ga preberemo iz datoteke. Postopek je podoben kot pri ostalih modulih zbirke OpenOffice.org.

Preglednice lahko dodamo tudi brez vstavljanja predmeta OLE: v



Slika 8.9: Vstavljanje in spreminjanje grafa v predstavitvi

pogovornem oknu za novo stran izberemo tisto obliko, ki ima tudi okvir s preglednico.

Podobno lahko vnesemo tudi 2D ali 3D graf. V pogovornem oknu izberemo obliko strani, ki ima okvir za grafikon. Ob kliku na okvir se prikaže privzeti graf. Spreminjamo ga lahko z uporabo glavne orodjarne, ki se prikaže ob levi strani okna (slika 8.9).

Podobno vstavljamo tudi grafiko. Seveda lahko dodajamo tudi ostale objekte, ki so dostopni preko izbire “Vstavi” v glavnem meniju.

Povezave na druge dokumente, ki so lahko tudi na spletu, pa vstavimo z izbiro “Vstavi→Hiperpovezava”. V vnosno polje “Cilj” moramo le vpisati URL (*Uniform Resource Locator*) dokumenta.

8.1.5 Prehodi in učinki

Prehode med stranmi lahko animiramo. Na voljo imamo preko 50 posebnih učinkov. Če nismo načina prehoda določili enotno za vse strani že pri ustvarjanju predstavitve s čarovnikom, moramo to sedaj storiti za vsako stran posebej.

Za določitev prehoda in posebnih učinkov odpremo pogovorno okno *Prehod med diapozitivi* z izbiro ukaza "Diaprojekcija→Prehod med diapozitivi". Nato izberemo stran, ki jo hočemo obogatiti s prehodom. Okno *Prehod med diapozitivi* je medtem ostalo odprto. V njem sedaj izberemo želeni učinek. Ko izberemo še hitrost, s klikom na zeleno kljukico učinek dodelimo izbrani strani. Gumb "Predogled" nam pomaga pri ogledu učinka. Učinke tako priredimo vsaki strani posebej. Najlažje to storimo v pogledu diapozitiva, tam lahko izberemo več strani, ki jim lahko določimo isti prehod.

V oknu *Prehod med diapozitivi* lahko nastavimo še nekaj možnosti (gumb "Dodatki"), med katerimi velja omeniti dodajanje zvočnih učinkov in omogočanje avtomatskega časovnega preklopa strani.

Učinki na objektih

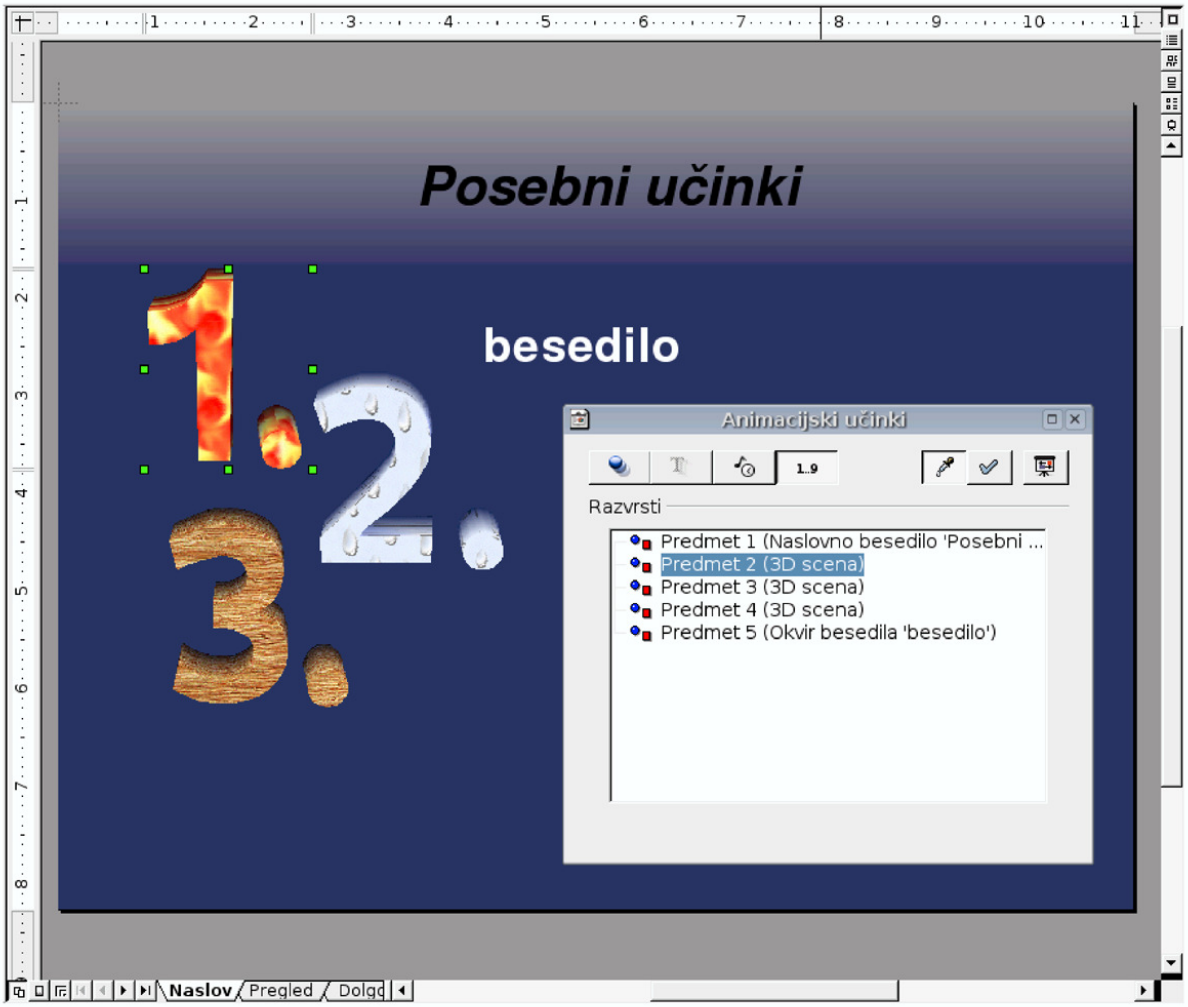
Vsem objektom v predstavitvi lahko nastavimo posebne učinke. Učinke nastavljamo v oknu *Animacijski učinki* ("Diaprojekcija→Učinki") (slika 8.10).

Okno ima štiri podokna; v prvem (*Učinki*) objektu določimo vrsto učinka. Izbiramo lahko med različnimi prikazi, zabrisi, raztegi, vrtenji itd. Ko učinek izberemo, kliknemo na zeleno kljukico. V drugem podoknu priredimo učinke besedilu; v tretjem podoknu (*Dodatki*) pa priredimo učinku ustrezne časovne parametre, določimo vidljivost in dodamo zvočne učinke. V podoknu *Vrstni red* vidimo vse objekte na strani, ki so kakorkoli animirani. V tem oknu nastavimo sosledje animacij.

Interaktivnost

Z izbiro "Diaprojekcija→Interakcija" lahko za izbrani objekt določimo akcijo, ki se bo izvršila ob kliku z miško na objekt med samo predstavitvijo (slika 8.11). Poglejmo nekaj primerov akcij:

- Zabriši predmet. Ob kliku se objekt animira.
- Skok na stran v predstavitvi. Objekt tako predstavlja kazalec, preko katerega lahko spremenimo tok govora.
- Pojdi v dokument: odpiranje poljubnega dokumenta.

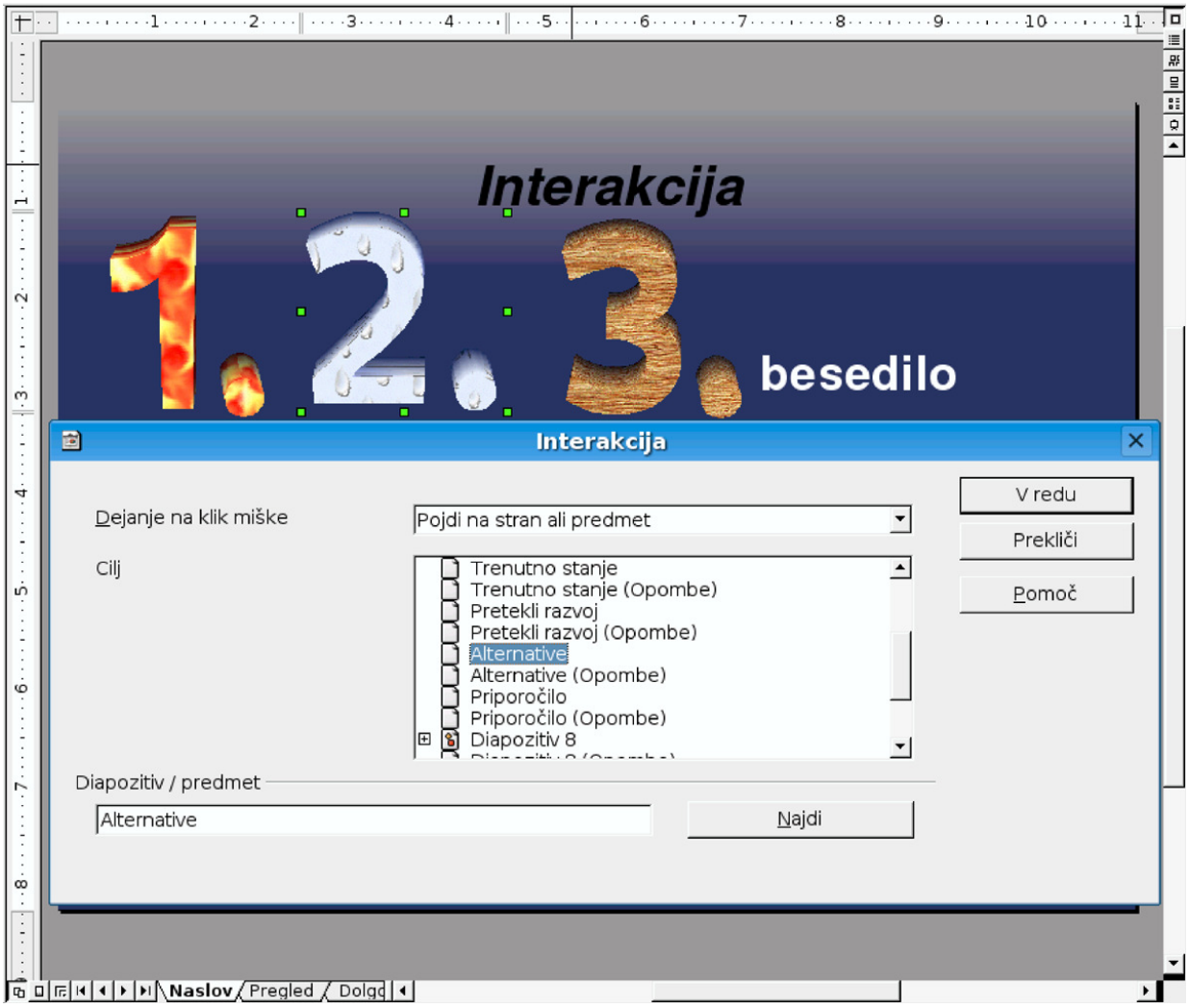


Slika 8.10: Okno za določanje učinkov

- Akcija nad samim objektom, na primer odpiranje preglednice.
- Zagon programa ali makroja.
- Predvajanje zvoka.

8.1.6 Izvedba predstavitve

Predstavitev poženemo z gumbom “Diaprojekcija” v glavni orodni vrstici ali z izbiro “Diaprojekcija→Diaprojekcija”. Pred javnim nastopom lahko preverimo dolžino in potek predstavitve s “Diaprojekcija→Preizkusi časovno usklajenost”. Končne nastavitve spremenimo v



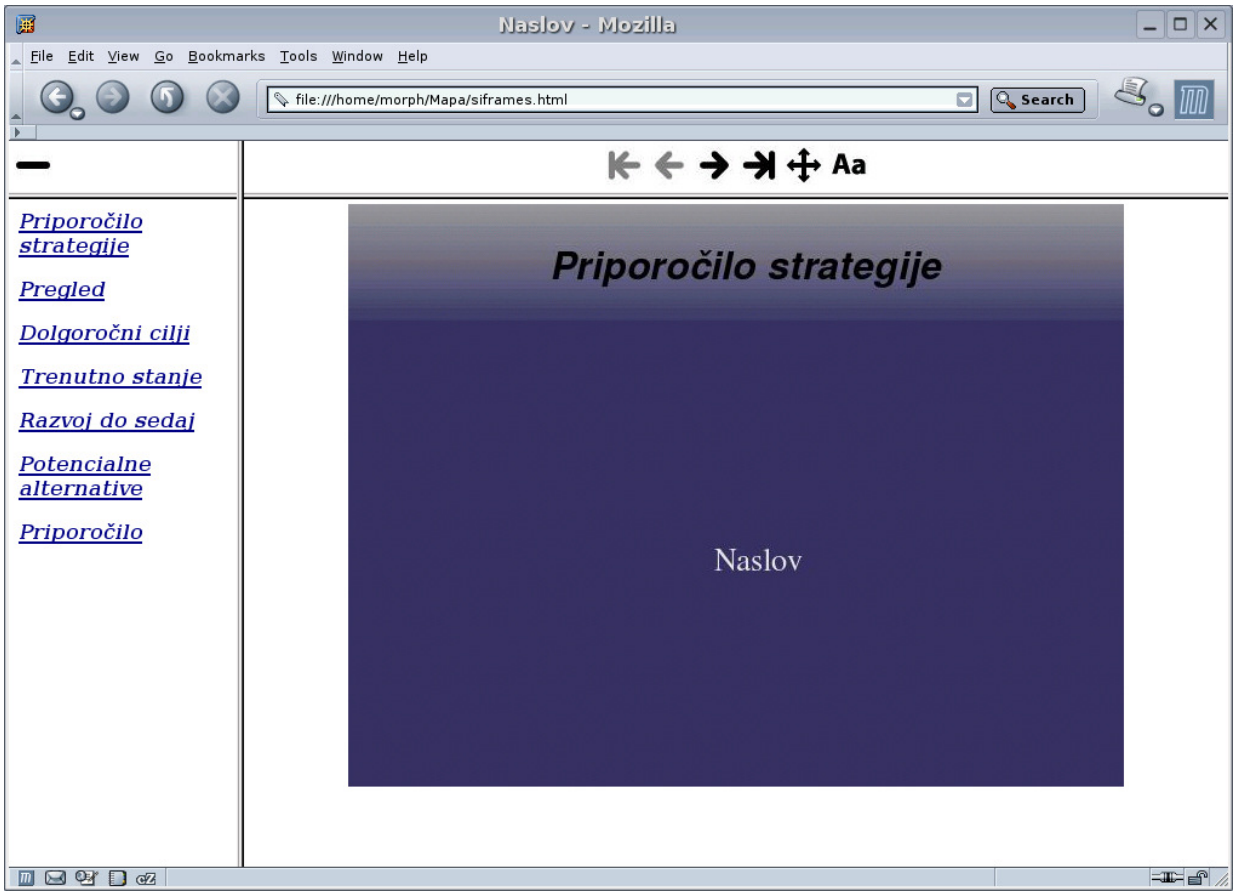
Slika 8.11: Interaktivno okno

"Diaprojekcija→Nastavitve diaprojekcije". V "Diaprojekcija→Diaprojekcija po meri" lahko iz obstoječe predstavitve izdelamo novo z vključevanjem in izključevanjem strani.

Predstavitev lahko izvozimo tudi v zapis HTML ali Macromedia Flash in jo tako predstavimo na spletu (slika 8.12). To storimo tako, da v izbiri "Datoteka→Izvozi" izberemo HTML oziroma SWF kot izhodni zapis. Predstavitev pa lahko izvozimo tudi v format PDF, tako da izberemo "Datoteka→Izvozi v PDF".

8.1.7 Naloge

1. Kaj moramo upoštevati pri pripravi govorne predstavitve?



Slika 8.12: Predstavitev na spletu

2. Naštejte katere vrste gradiv moramo pripraviti za govorno predstavitev.
3. Na kakšne načine lahko prikažemo projekcijsko gradivo? Kateri način se največkrat uporablja?
4. Koliko prosojnic lahko največ pripravimo za 10 minutno predstavitev?
5. V katere različne zapise lahko shranimo končno predstavitev?
6. V obstoječo predstavitev dodajte novo stran s predmetom OLE.
7. V obstoječi predstavitvi spremenite vrstni red strani in prvo stran skrijte.
8. Spremenite izgled celotne predstavitve, tako da dodate pripadajočo številko strani.
9. Med prvo in drugo stran dodajte nov prehod.

10. Dodajte nov objekt iz galerije in določite, da izgine, ko kliknemo nanj.

8.2 Koristne spletne povezave

1. Spletna stran OpenOffice.org:
   `http://www.openoffice.org/`
2. Spletna stran slovenskega OpenOffice.org:
   `http://sl.openoffice.org/`
   (S te strani je pod licenco GNU/FDL dostopna tudi knjiga [4].)
3. Zavezništvo OpenOffice.org – koristne informacije glede namestitve in uporabe paketa OpenOffice.org:
   `http://ooz.agenda.si/`
4. Adobe Portable Document Format (PDF):
   `http://www.adobe.com/products/acrobat/adobepdf.html`

8.3 Literatura

[1] J. A. Devito. *Human Communication, The Basic Course.* Longman, New York, NY, seventh edition, 1997.

[2] S. Haugland and F. Jones. *OpenOffice.org 1.0 Resource Kit.* Prentice-Hall PTR, Upper Saddle River, NJ, 2003.

[3] G. Leete, E. Finkelstein, and M. Leete. *OpenOffice.org for Dummies.* Wiley Publishing, Inc., Hoboken, NJ, 2003.

[4] R. Ludvik, I. Zajc, and A. Medic. *Hitri vodnik po OpenOffice.org.* Pasadena, Ljubljana, 2003. (Dostopna pod licenco GNU/FDL na: http://openoffice.lugos.si/knjiga/Hitri_vodnik_po_OpenOffice.org_FDL.pdf)

9

Matematični programi

Med uporabniško programsko opremo najdemo različne programe in programske pakete, ki so namenjeni reševanju matematičnih problemov. Med njimi so najbolj znani *Mathematica*, *Matlab*, *Maple* in *Octave*. Poleg njih najdemo na trgu tudi manj obsežne in zmogljive programe, kot je na primer *Derive*. Programske pakete lahko delimo glede na osnovni namen; tako je Matlab predvsem namenjem numeričnemu reševanju problemov, pri programskih paketih Mathematica in Maple pa je poudarek predvsem na simboličnem reševanju problemov. Podobnih programov je sicer še več, vsak pa ima svoje prednosti in slabosti.

Vsa ta programska orodja omogočajo več načinov uporabe. Najenostavnejši način je interaktivno delovanje, pri katerem uporabnik vpiše ukaz ali zbirko ukazov, sistem pa jih izvede in sporoči rezultat. Poleg tega imajo programska orodja vgrajene programske jezike, s katerimi je možno učinkoviteje izkoristiti obstoječe funkcije ali pa celo razširiti funkcionalnost paketa z dodatnimi novimi funkcijami. Ta odprtost je povzročila, da danes obstaja veliko zbirk namenskih razširitvenih paketov. Prav tako je možno programe združevati in dopolnjevati s programi, napisanimi v standardnih programskih jezikih, kot sta na primer C in FORTRAN. Vse te programske pakete odlikuje tudi zmogljiv grafični vmesnik, ki omogoča vizualizacijo poteka rešitev ter končnih rezultatov.

Velika razširjenost matematičnih programskih orodij, ki so na voljo za večino najbolj znanih operacijskih sistemov in tečejo na vseh vrstah računalnikov, od osebnih računalnikov do superračunalnikov, je prispevala k uporabi na vrsti področij:

- učenje matematike (koncepti, primerjalne študije),
- učenje in raziskave na področju tehnike, računalništva, fizike, avtomatike, ekonomije, kemije, biologije itd.,
- raziskave in razvoj v industriji.

V tem poglavju se bomo seznanili z dvema najbolj razširjenima matematičnima programoma; Mathematico, ki je namenjena v predvsem simbolnemu reševanju matematičnih problemov, in Matlabu, ki je numerično usmerjeno orodje.

9.1 Mathematica

Mathematica je učinkovit, morda celo revolucionaren programski paket oziroma jezik, ki omogoča integracijo simboličnih, numeričnih in grafičnih funkcij. S svojim uporabniškim vmesnikom omogoča ustvarjanje dokumentov, ki so na las podobni besedilom, ki so nam že zlezli pod kožo: knjige, seminarske naloge, ipd. Vendar nam ti dokumenti omogočajo tudi dinamično prevajanje, saj so v osnovi sestavljeni iz živih enačb in animacij. Žive enačbe so v bistvu celice dokumenta, ki jih Mathematica ovrednoti in izpiše rezultat pod ustrezno celico.

Mathematica ima konsistentno sintakso in sledi začrtanim razvojnim principom. To omogoča hitro in enostavno učenje Mathematice, hkrati pa ovrže krivično trditev, da je učenje Mathematice težavno.

V podjetju Wolfram Research, kjer so naredili Mathematico, pravijo, da je Mathematica sistem za izvajanje matematike na računalniku. Dejstvo pa je, da je Mathematica paket, ki je uporaben na vseh tehniških področjih.

Na trgu je že peta različica Mathematice, vendar naša predstavitev velja tudi za starejše različice oziroma vsaj za tretjo različico. Osnovna datoteka Mathematice ima podaljšek `nb`, na primer `vaja.nb`, pri čemer `nb` izhaja iz angleške besede *notebook* – beležnica.

Mathematica je plačljiv program, zato ga žal na priloženi zgoščenki Slix ni. Različice Mathematice obstajajo za večino sodobnih računalniških platform (Linux, Macintosh, MS Windows, ...).

9.1.1 Mathematica pod Linuxom

Z Mathematico lahko pod Linuxom delamo na dva načina:

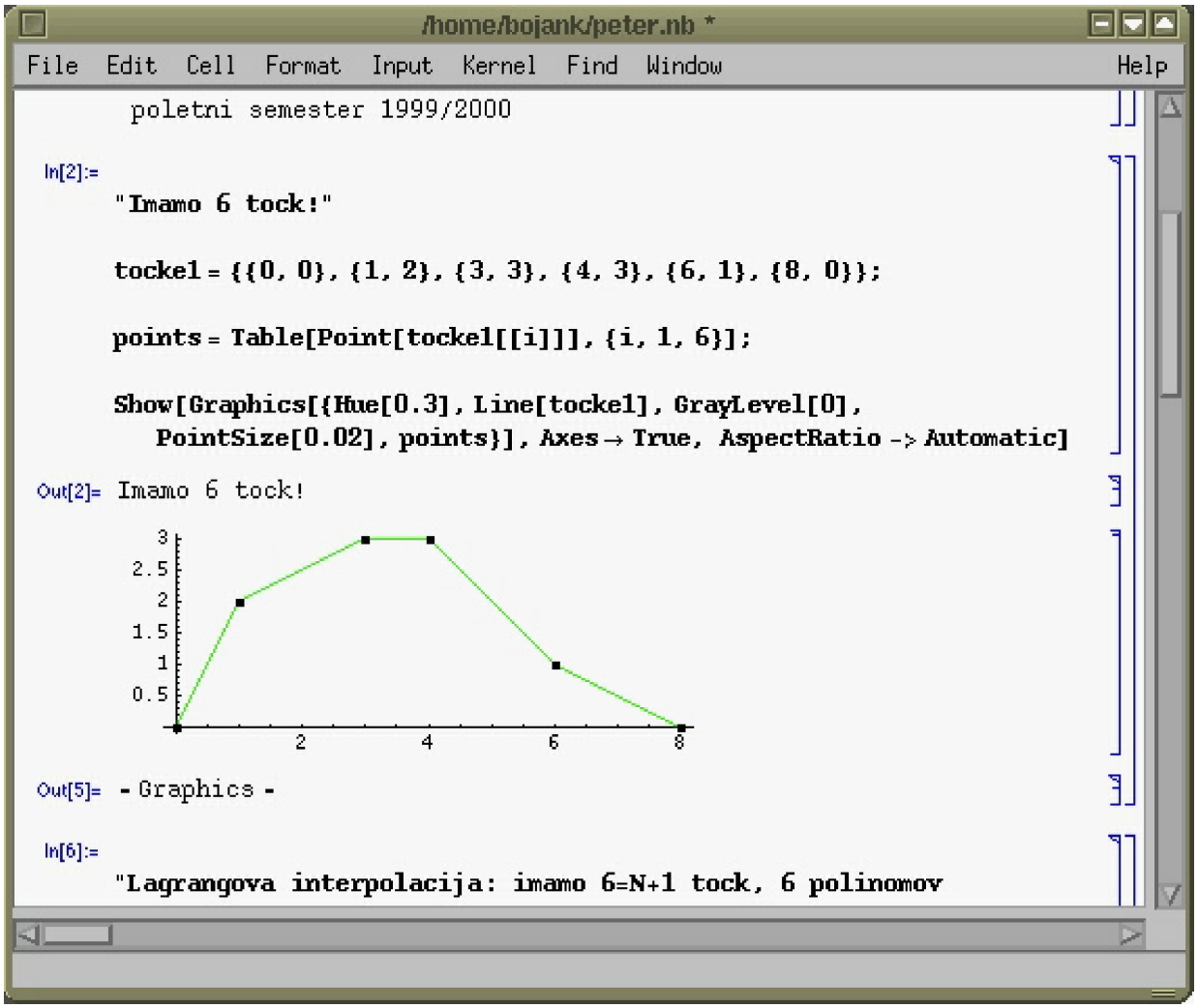
- v ukazni vrstici terminalskega okna ali
- v beležnici grafičnega vmesnika.

Delo v ukazni vrstici pričnemo z ukazom `math`, ki ga izvršimo v terminalskem oknu. Če pa v terminalskem oknu izvršimo ukaz `mathematica`, se zažene grafična različica programa, kjer vhod programu podajamo v beležnici programa. Razlika je torej v uporabniškem vmesniku. Poleg samega uporabniškega vmesnika pa ima Mathematica

tudi del, ki so ga avtorji poimenovali jedro (angl. *Kernel*). Ta skrbi za izvajanje vhodne kode in je za uporabnika praktično neviden.

Uporaba beležnice

Ker so grafični vmesniki do uporabnikov bolj prijazni, bomo v tem poglavju delali v načinu beležnice, čeprav ni bistvenih razlik v delovanju med obema omenjenima načinoma. Slika 9.1 prikazuje grafični uporabniški vmesnik programa Mathematica. Iz slike je lepo razvidno, da je program sestavljen iz dveh delov: menijskega dela in beležnice.



Slika 9.1: Grafični uporabniški vmesnik programa Mathematica

Menijski del je dokaj standarden: če na primer želimo ustvariti novo datoteko s končnico `nb`, potem izberemo meni "File" in opcijo "New", če

želimo datoteko shraniti, potem izberemo meni "File" in opcijo "Save" itd. Hkrati pa nam menijski del omogoča tudi celo vrsto drugih operacij. Oglejmo si nekaj zanimivih, predvsem s stališča povezave z drugimi programskimi orodji:

- Zapišimo postopek shranjevanja ustvarjene slike v format EPS (Encapsulated PostScript), ki je med drugim osnovni format slik za vključevanje le-teh v dokumente logičnega urejevalnika besedil LaTeX: označimo želeno sliko, izberemo meni "Edit", opcijo "Save Selection As", znotraj nje opcijo "EPS", vpišemo želeno ime slike in jo shranimo na želeno mesto.

- Kako shranimo datoteko s končnico `nb` kot dokument urejevalnika LaTeX? Izberemo meni "File", opcijo "Save As Special", znotraj nje opcijo "TeX", vpišemo želeno ime dokumenta in ga shranimo na želeno mesto. Mathematica ustvari osnovno datoteko, katere zapis ustreza sintaksi sistema LaTeX in pripadajoče slike, če te v datoteki s končnico `nb` obstajajo. Preden prevedemo ustvarjeno datoteko LaTeX pa moramo poskrbeti, da prevajalnik LaTeX najde ustrezno stilno datoteko, ki je priložena Mathematici. Ime te stilne datoteke in kje se le-ta nahaja, je zapisano v glavi ustvarjene datoteke LaTeX.

- Kako shranimo datoteko s končnico `nb` kot dokument hipertekstovnega jezika HTML? Izberemo meni "File", opcijo "Save As Special", znotraj nje opcijo "HTML", vpišemo želeno ime dokumenta in ga shranimo na želeno mesto. Torej podobno kot zgoraj, le da sedaj izberemo opcijo "HTML" namesto opcije "TeX". Tudi tukaj Mathematica ustvari osnovno datoteko, katere zapis sedaj ustreza sintaksi jezika HTML, vendar je tukaj vsaka celica datoteke s končnico `nb` predstavljena s sliko v formatu GIF.

V beležnici lahko vidimo kar nekaj njenih značilnosti:

- Na levi strani so vsi vhodi označeni z besedico *In*, vsi izhodi pa z besedico *Out*, hkrati pa Mathematica vhode in izhode tudi številči. Številčenje nam pride prav pri sklicu na določen izraz, kar pomeni, da lahko namesto celotnega izraza v kodi podamo le ustrezno številko izraza, ki smo ga že ovrednotili. O tem bomo več spregovorili v razdelku 9.1.3, na tem mestu pa izpostavimo le še dejstvo, da besedici *In* in *Out* s pripadajočo številko ustvari Mathematica sama.

- Na desni strani so označene posamezne celice.

- Vhod je zapisan s krepkimi črkami, izhod pa z običajnimi.

Med celicami in po celicah se pomikamo s pomočjo smernih tipk, dodajanje novih celic pa je možno le v primeru, ko se v beležnici na mestu, kjer želimo celico dodati, pojavi horizontalna črta.

Vrednotenje vhoda

Vrednotenje izrazov, ki jih Mathematici podamo na vhodu, poteka na tri načine:

- Če želimo ovrednotiti le posamezno celico beležnice, se postavimo v to celico in pritisnemo `<Shift>+<Enter>`. Zgolj pritisk na tipko `<Enter>` nam povzroči skok v novo vrstico beležnice.
- Če želimo ovrednotiti več celic naenkrat, jih moramo najprej označiti. To naredimo tako, da z miško klikamo na želene celice, hkrati pa držimo pritisnjeno tipko `<Control>`. Sam postopek ovrednotenja tako označenih celic je enak kot zgoraj.
- Če želimo ovrednotiti celotno beležnico, moramo iz menija "Kernel" izbrati opcijo "Evaluation" in znotraj nje opcijo "Evaluate notebook".

9.1.2 Osnovna notacija Mathematice

V tem poglavju bomo izpostavili nekaj osnovnih razlik med sintakso Mathematice in navadno matematično notacijo.

Osnovne operacije

Podajmo osnovne operacije in njihove simbole v Mathematici v obliki tabele:

Osnovne operacije	*Običajna matematična notacija*	*Notacija v Mathematici*
seštevanje	$n + m$	`n+m`
odštevanje	$n - m$	`n-m`
množenje	$n * m$	`n*m` ali s presledkom `n m`
deljenje	$\frac{n}{m}$	`n/m`
potenciranje	n^m	`n^m`
skalarni produkt	$\vec{\mathbf{m}} \cdot \vec{\mathbf{n}}$	`m.n`

Pri množenju obstaja tudi nekaj izjem. Za ilustracijo si oglejmo naslednji primer:

```
2 x = 2x = 2*x = 2(x).
```

Zaradi tega ne moremo začeti imena spremenljivke s številko. Velja tudi, da do napake ne pride, če uporabimo presledke na mestih, kjer logično ne more biti množenja. Na primer: `a+b = a + b`.

Ostale operacije niso znakovnega značaja, pač pa funkcijskega značaja, zato so obravnavane v razdelku 9.1.3, kjer bo tekla beseda o vgrajenih funkcijah in njihovi uporabi.

Pomen različnih oklepajev

Sintaksa Mathematice se od običajne matematične notacije razlikuje v dveh (od treh) načinih uporabe oklepajev:

Uporaba oklepajev	*Običajna matematična notacija*	*Notacija v Mathematici*
grupiranje	$a * (b + c)$	`a*(b+c)`
točka v ravnini (seznam)	(p, q)	`{p,q}`
funkcija f v odvisnosti od spremenljivke t	$f(t)$	`f[t]`

Definiranje funkcij

V Mathematici definiramo funkcijo s pomočjo niza `:=` in ne zgolj z enačajem. Če želimo v Mathematici definirati funkcijo f, ki pripiše neko realno število neodvisni spremenljivki t, to zapišemo kot:

`f[t_]:=` nek izraz z neodvisno spremenljivko t

Tukaj je podčrtaj, ki sledi črki `t`, s stališča sintakse Mathematice bistvenega pomena: `t` pomeni neodvisno spremenljivko t, `t_` pa pomeni generični t.

Uporaba enakosti

V Mathematici poznamo več simbolov za enačenje, njihova uporaba pa je odvisna od namena enačenja:

Uporaba enakosti	*Običajna matematična notacija*	*Notacija v Mathematici*
prireditev	$a = 5$	`a=5`
reševanje enačb	$2x + 3y = 5$	`2*x+3*y==5`

Tudi definiranje funkcij, o katerem smo spregovorili v prejšnjem razdelku, bi lahko obravnavali kot poseben primer enačenja.

Tipične napake

Pri delu z Mathematico pride velikokrat do neljubih napak, še posebej pri začetnikih, zato na kratko navedimo tipične napake uporabnikov, zaradi katerih program ne deluje pravilno:

- Opazili smo, da nekateri uporabniki sami vpisujejo že omenjeni besedici `In` in `Out`, kar je narobe. Za vse to poskrbi Mathematica sama!

- Velikokrat se zgodi, da se Mathematica zaradi kompliciranih oziroma nepravilnih izrazov "obesi". V takšnem primeru lahko poskusimo problem rešiti na enega od sledečih načinov: pritisnemo tipki `<Alt>` in piko (`<.>`), izvajanje poskusimo prekiniti iz jedra (meni "Kernel"), poskusimo shraniti trenuten dokument in zapreti Mathematico, poskusimo zapreti Mathematico s pomočjo sistemskih ukazov ali pa, v skrajnem primeru, ko vse drugo odpove, ponovno zaženemo računalnik.

- V primerih, ko ima neka spremenljivka že podano vrednost, mi pa želimo uporabiti isto spremenljivko brez vrednosti, moramo najprej izbrisati njeno vrednost. To naredimo z ukazom (s funkcijo) `Clear` ali `Remove`, kjer med oglatima oklepajema `[]` podamo imena spremenljivk, ki jim želimo izbrisati njihove vrednosti.

- Napačna uporaba enakosti.

- Napačna uporaba oklepajev.

- Napačna uporaba množenja.

- Do napake lahko pride tudi zaradi klica neke funkcije, ki je Mathematici dosegljiva le preko predhodnega klica paketa, ki to funkcijo vsebuje. Če to funkcijo uporabimo pred dejanskim klicem paketa, največkrat zadostuje, če nad to funkcijo uporabimo omenjeni ukaz `Remove`. Če to ne pomaga, je najbolje, da zapremo jedro, kar naredimo tako, da izberemo meni "Kernel", opcijo "Quit Kernel" in znotraj nje opcijo "Local", nato pa naložimo paket pred dejansko uporabo funkcije iz tega paketa. O nalagaju paketov bomo spregovorili na koncu razdelka 9.1.3.

Posebne oznake

Podobno kot LaTeX ima tudi Mathematica nekaj posebnih oznak, ki imajo znotraj kode poseben pomen. Oglejmo si osnovne oznake in njihov pomen:

Oznaka	*Pomen*
(*	začetek komentarja
*)	konec komentarja
″	začetek in konec spremnega besedila
\n	skok v novo vrstico
#	ekvivalent ukazu `Slot`
&	zaključni znak definicije funkcije

Ker uporaba teh oznak presega osnove znanja o Mathematici, ki so podane v tem poglavju, ne bomo podajali primerov uporabe, bralec pa si lahko več o posebnih oznakah ogleda v [2, 3] ali kakšni podobni knjigi, ki obravnava Mathematico.

Osnovni izračuni

Začnimo dejansko učenje Mathematice z nečim že znanim: aritmetiko. Za začetek uporabimo Mathematico kot kalkulator:

```
In:=  3+7
Out=  10
```

Vedno je pametno, da izraze v Mathematici poimenujemo, saj se velikokrat v nadaljevanju sklicujemo nanje. Kot omenjeno, se lahko na izraze sklicujemo tudi na podlagi prirejenih številk (razdelek 9.1.3), vendar se to v praksi ne obnese najbolje, saj se lahko zgodi, da boste nekje v prihodnosti zagnali le del napisane kode, to pa povzroči ponovno številčenje (le) prevedenega dela. Zato poimenujmo rezultat z `a`:

```
In:=  a=3+7
Out=  10
```

Če sedaj vprašamo po `a`, dobimo vrednost `10`:

```
In:=  a
Out=  10
```

Če vprašamo po `a+b`, dobimo vrednost `10+b`, saj `b` trenutno še nima prirejene vrednosti:

```
In:=  c=a+b
Out=  10+b
```

Če sedaj spremenljivki `b` pripišemo vrednost `5`, dobi `c` vrednost `15`:

```
In:=  b=5
Out=  5
In:=  c
Out=  15
```

Vrednost spremenljivke `b` lahko sedaj brišemo s pomočjo omenjenega ukaza `Clear`:

```
In:=  Clear[b]
      c
Out=  10+b
```

Priredimo spremenljivki `b` vrednost $\frac{11}{3}$:

```
In:=  b=11/3
Out=  11/3
In:=  c
Out=  41/3
```

Opazimo lahko, da je Mathematica zapisala obe števili kot ulomek in ne kot decimalno število. Mathematica namreč vedno poskuša vzdrževati točnost, razen v primeru, ko tega eksplicitno ne želimo. Zato Mathematica vrednost $\frac{11}{3}$ zapiše kot ulomek, saj bi vsako decimalno število bilo zgolj približek te vrednosti.

Oglejmo si, kako Mathematico pripravimo do tega, da nam kot rezultat vrne decimalno število:

```
In:=  N[c]
Out=  13.6667
In:=  Pi
Out=  π
In:=  N[Pi]
Out=  3.14159
```

Na tem mestu smo uporabili vgrajeno funkcijo `N`, ki vrne numerično vrednost izraza v oglatih oklepajih `[]`. Celoten izraz za `N` je `N[izraz,n]`, ki izračuna izraz na `n` števil natančno[1]. Če vrednosti spremenljivke `n` ne podamo, Mathematica privzame število 6.

[1] `n` podaja skupno število števk v celem in decimalnem delu.

Tudi aritmetične operacije s kompleksnimi števili so skoraj tako enostavne kot aritmetične operacije z realnimi števili. V Mathematici označimo kvadratni koren iz -1 z `I`. Spodnji primer podaja seštevanje, množenje in deljenje kompleksnih števil. Z vgrajeno funkcijo `Abs` izračunamo absolutno vrednost, s funkcijo `Conjugate` konjugirano vrednost, funkciji `Re` in `Im` pa nam podata realni in imaginarni del kompleksnega števila.

```
In:=  (-1)^(1/2)
Out=  I
In:=  a=4+5I
Out=  4+5I
In:=  b=6+7I
Out=  6+7I
In:=  a+b
Out=  10+12I
In:=  a b
Out=  -11+58I
In:=  Abs[a b]
Out=  √3485
In:=  a/b
Out=  59/85 + 2I/85
In:=  Re[a/b]
Out=  59/85
In:=  Im[a/b]
Out=  2/85
In:=  Conjugate[a]
Out=  4-5I
```

Pri deljenju vhoda na več vrstic moramo biti zelo pozorni. Ilustrirajmo to na primeru:

```
In:=  a=7+10
      +317
Out=  17
      317
In:=  a
Out=  17
```

Tega seveda nismo pričakovali. Zakaj ima `a` vrednost 17 in ne 334? Mathematica namreč ovrednoti vsako vrstico vhoda ločeno, razen če le-ta ne predstavlja sintaktično pravilnega stavka Mathematice. V spodnjem primeru smo `a` zapisali nekoliko drugače – če razdelimo vrstico za znakom plus, dobimo, kar smo hoteli:

```
In:=  a=7+10+
      317
Out=  334
In:=  a
Out=  334
```

Za konec tega razdelka pa si oglejmo še eno zanimivost delovanja Mathematice, ki je sicer logična, a kljub temu presenetljiva. Če sedaj vprašamo po vrednosti spremenljivke `c`, dobimo, glede na kodo, zapisano v tem poglavju, naslednji rezultat:

```
In:=  c
Out=  16+7I
```

Spremenljivko `c` smo sicer res definirali kot `a+b`, vendar ima sedaj vrednost `10+b`, saj smo pred spremenljivko `c` definirali spremenljivko `a` in ji pripisali vrednost `3+7`. Spremenljivka `c` je torej v tem trenutku odvisna zgolj od spremenljivke `b`. Če pa bi najprej definirali `c=a+b` in nato spremenljivko `a=3+7`, bi bila spremenljivka `c` odvisna od spremenljivk `a` in `b`.

Mathematica je torej zelo občutljiva na zaporedje vpisovanja vhoda in zaporedje prevajanja celic!

9.1.3 Uporaba vgrajenih funkcij

Kot smo lahko opazili, se vse vgrajene funkcije v Mathematici začnejo z veliko črko. Če zahtevajo tudi argumente, jih zapišemo med oglate oklepaje `[]`. Na tem mestu naj ponovimo tudi uporabo ostalih oklepajev: okrogle oklepaje uporabljamo za določitev vrstnega reda operacij, zavite oklepaje pa za sezname, na primer točke v treh dimenzijah.

Vse vgrajene funkcije imajo polna imena, torej niso okrajšave, saj si jih tako lažje zapomnimo. V prid temu govori tudi dejstvo, da ima Mathematica okoli 1000 funkcij.

Če recimo pozabimo sintakso ali uporabo nekega ukaza, lahko Mathematico vprašamo po njegovi sintaksi oziroma uporabi enostavno tako, da pred njegovo ime dodamo vprašaj (`?`).

Primer:

```
In:=  ?Plot3D
Out=  Plot3D[f, {x, xmin, xmax}, {y, ymin, ymax}]
      generates a three-dimensional plot of f as a
      function of x and y.
      Plot3D[{f, s}, {x, xmin, xmax}, {y, ymin, ymax}]
      generates a three-dimensional plot in which the
      height of the surface is specified by f, and
      the shading is specified by s.
```

Če pred ukaz dodamo dvojni vprašaj (`??`), pa nam Mathematica postreže s še več podatki o ukazu.

Tudi v primeru, da pozabite del imena ukaza, lahko Mathematico vprašate po ukazih, ki vsebujejo del, za katerega ste prepričani, da je del celotnega ukaza. Za ilustracijo si oglejmo naslednji *primer*:

```
In:=  ?*Pol*
```

V tem primeru nam bo Mathematica izpisala vse ukaze, ki vsebujejo v svojem imenu del `Pol`.

Trigonometrične funkcije

Sintaksa funkcije za sinus kota je `Sin[x]`, za kosinus `Cos[x]`, za tangens `Tan[x]`, za kotangens `Cot[x]`, za sekans `Sec[x]` in za kosekans `Csc[x]`. Mathematica predpostavlja, da je kot `x` podan v radianih.

Primer:

Izračunajte sinus kota $\pi/2$!

```
In:=  Sin[Pi/2]
Out=  1
```

Korenjenje

Ukaz za kvadratni koren je `Sqrt[x]`. Mathematica ukaz interpretira kot izraz $x^{1/2}$.

Primer:

Izračunajte izraz: $\sqrt{4} + 27^{1/3}$!

```
In:=  Sqrt[4]+27^(1/3)
Out=  5
```

Logaritmiranje

Ukaz za logaritmiranje je `Log[x]` oziroma `Log[o,x]`. Prvi ukaz predvideva, da gre za naravni logaritem, torej logaritem z osnovo e (število e v Mathematici označimo z `E`), pri drugem pa osnovo podamo s parametrom `o`.

Primer:

Izračunajte logaritem števila 8 pri osnovi 2!

```
In:=  Log[2,8]
Out=  3
```

Sklicevanje

Kot omenjeno, se lahko na že ovrednotene celice sklicujemo. Ukaz za sklic na vhodno vrstico je `In[x]`, ukaz za sklic na izhodno vrstico pa `Out[x]`. `x` je v tem primeru zaporedna številka celice, ki jo dodeli Mathematica. Na tem mestu pa ponovno opozorimo, da Mathematica ob naknadnem prevajanju dela kode, prevedene celice oštevilči drugače (stran 254).

Primer:

```
In[19]:=  Sqrt[4]
Out[19]=  2
In[20]:=  Log[2, 8]
Out[20]=  3
In[21]:=  In[20] + Out[19]
Out[21]=  5
```

Risanje

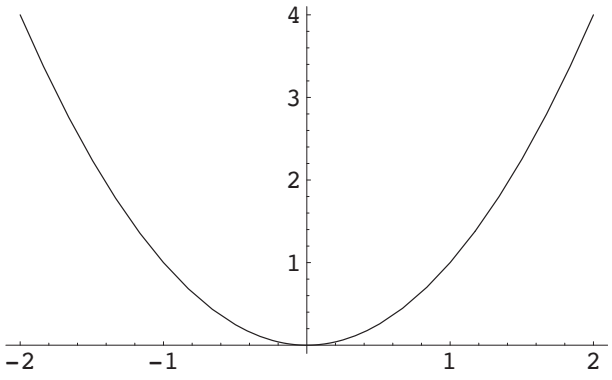
Oglejmo si dve bistveni vgrajeni funkciji za risanje v dvorazsežnem prostoru:

1. Ukaz `Plot` je namenjen risanju eksplicitno podanih funkcij, torej funkcij oblike $y = f(x)$, in ima naslednjo (osnovno) sintakso: `Plot[funkcija,{x,spodnja_meja,zgornja_meja}]`.

   Primer:

   Narišite funkcijo x^2 na intervalu od -2 do 2!

```
In:= Plot[x^2,{x,-2,2}]
```



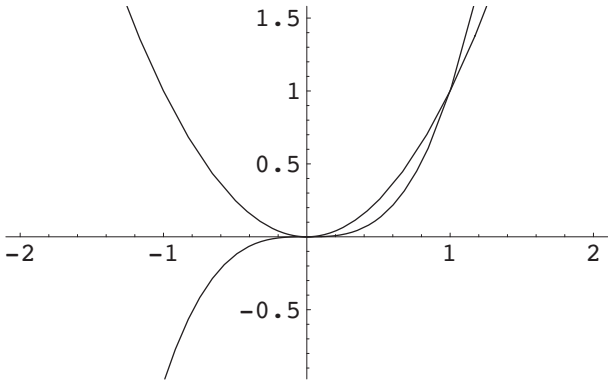

```
Out=
```

   Na enem grafu pa lahko narišemo tudi več funkcij naenkrat. Razširjena sintaksa ima sedaj naslednji izgled:
   `Plot[{funkcija1,funkcija2,...},`
   `{x,spodnja_meja,zgornja_meja}]`.

   Primer:

   Narišite funkciji x^2 in x^3 na intervalu od -2 do 2!

```
In:= Plot[{x^2,x^3},{x,-2,2}]
```




```
Out=
```

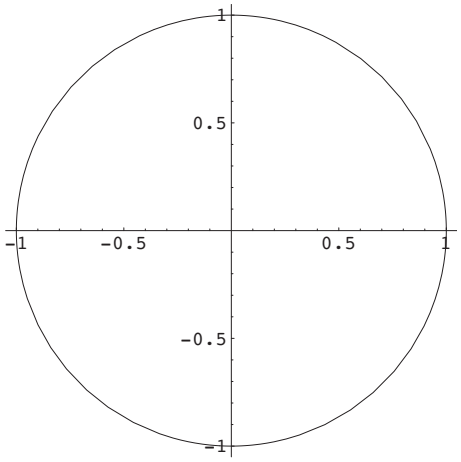
2. Ukaz `ParametricPlot` je namenjen risanju parametrično podanih funkcij[2]. Njegova (osnovna) sintaksa se glasi:
`ParametricPlot[{fx,fy},{t,spodnja_meja,zgornja_meja}]`.

Primer:

Narišite funkcijo $\{\cos(t), \sin(t)\}$ na intervalu od 0 do 2π!

```
In:= ParametricPlot[{Cos[t], Sin[t]}, {t, 0, 2Pi},
     AspectRatio->Automatic]
```

Out=



Ukaz `AspectRatio->Automatic` zagotovi enakomerno razdelitev osi na enote.

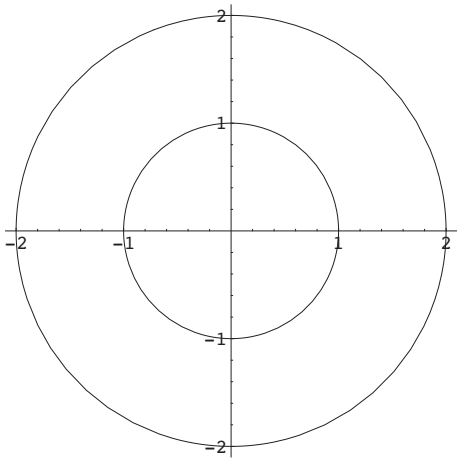
Podobno kot pri ukazu `Plot`, lahko tudi pri ukazu `ParametricPlot` na enem grafu prikažemo več funkcij. Oglejmo si razširjeno sintakso:
`ParametricPlot[{{fx1,fy1},{fx2,fy2},...},`
`{t,spodnja_meja,zgornja_meja}]`.

Primer:

Narišite funkciji $\{\cos(t), \sin(t)\}$ in $\{2\cos(t), 2\sin(t)\}$ na intervalu od 0 do 2π!

[2]Funkcija je podana parametrično, če sta medsebojno prirejeni vrednosti spremenljivk x in y izraženi s tretjo spremenljivko, imenovano parameter.

```
In:=  ParametricPlot[{{Cos[t], Sin[t]},{2Cos[t],
      2Sin[t]}}, {t, 0, 2Pi},
      AspectRatio->Automatic]
```




```
Out=
```

Obstajata tudi istoimenski funkciji s podaljškom 3D, ki sta, kot je iz imen očitno, namenjeni risanju v treh razsežnostih (`Plot3D`, `ParametricPlot3D`). Seveda pa obstaja še veliko drugih funkcij, ki so namenjene risanju: `ListPlot`, `ContourPlot`, `DensityPlot` itd.

Reševanje sistemov enačb

Za reševanje enačb in sistemov enačb uporabljamo ukaz `Solve`. Njegova (osnovna) sintaksa se glasi:
`Solve[enačbe, spremenljivke]`.

Primera:

Poiščite rešitvi za x iz enačbe $x^2 + 2bx + c = 0$!

```
In:=  Solve[x^2+2b x+c==0, x]
Out=  {{x→-b-√(b^2 - c)},{x→-b+√(b^2 - c)}}
```

Rešite sistem enačb: $x = 1 + 2ay$ in $y = 9 + 2x$!

```
In:=  Solve[{x==1+2a y, y==9+2x}, {x,y}]
Out=  {{x→-(1+18a)/(-1+4a)},{y→-11/(-1+4a)}}
```

Odvajanje

Oglejmo si sintakso ukaza za odvajanje `D` (velja tudi za parcialne odvode):

`D[funkcija,{spremenljivka,red_odvoda}]`.

Primer:

Poiščite četrti odvod izraza x^n!

```
In:=  D[x^n,{x,4}]
Out=  (-3+n) (-2+n) (-1+n) n x^(-4+n)
```

Integriranje

Ukaz `Integrate` ima naslednjo sintakso:

- `Integrate[funkcija,spremenljivka]` za nedoločen integral in
- `Integrate[f,{spremenljivka,spodnja_meja,zgornja_meja}]` za določen integral.

Primera:

Poiščite nedoločen integral izraza x^n!

```
In:=  Integrate[x^n,x]
Out=  x^(1+n)/(1+n)
```

Poiščite določen integral izraza x^3 na intervalu od 0 do 1!

```
In:=  Integrate[x^3,{x,0,1}]
Out=  1/4
```

Limita

Ukaz za izračun limit `Limit` ima naslednjo sintakso:
`Limit[izraz,x->x0]`.

Primer:

Poiščite limito izraza $\frac{\sin x}{x}$, ko gre x proti 0!

```
In:=  Limit[Sin[x]/x, x->0]
Out=  1
```

Matrike

Matrike definiramo kot seznam seznamov, kjer posamezne komponente ločimo med seboj z vejico (`,`). Kot že omenjeno, pa seznam definiramo med zavitimi oklepaji {}.

Primer:

Sestavite matriko $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$!

```
In:=  {{a, b}, {c, d}} // MatrixForm
Out=  ⎛ a  b ⎞
      ⎝ c  d ⎠
```

V zgornjem primeru je ukaz `MatrixForm`[3] uporabljen zgolj kot orodje za prikaz seznama v nam razumljivejši obliki.

Sedaj pa podajmo še osnovne operacije nad matrikami:

Operacija	*Ukaz*
produkt	`.` ali `Dot[spremenljivke]`
inverzna matrika	`Inverse[kvadratna_matrika]`
transponiranje	`Transpose[matrika]`
determinanta	`Det[kvadratna_matrika]`
potenciranje	`MatrixPower[kvadratna_matrika,stopnja]`

Operacija za produkt ima v primeru vektorjev seveda pomen skalarnega produkta.

Primeri uporabe operacij:

[3] Vhodna vrstica `{{a, b}, {c, d}} // MatrixForm` je ekvivalentna vrstici `MatrixForm[{{a, b}, {c, d}}]`!

```
In:=  a={1,2};
      b={3,4};
      A={a,b};
      B={b,a};
      a.b
      A.B
Out=  11
      {{5, 8}, {13, 20}}
In:=  Inverse[A]
Out=  {{-2, 1}, {3/2, -1/2}}
In:=  Transpose[{{1,2,3},{4,5,6}}]
Out=  {{1, 4},{2, 5},{3, 6}}
In:=  Det[A]
Out=  -2
In:=  MatrixPower[A,3]
Out=  {{37, 54}, {81, 118}}
```

V prvi vhodni celici lahko vidimo, da smo prve štiri izraze zaključili s podpičjem (;). Z njim dosežemo, da Mathematica v izhodni celici ne izpiše rezultata ovrednotenja izraza, ki je zakjučen s podpičjem.

Vektorski produkt

Zapišimo sintakso ukaza za vektorski produkt `Cross`:
`Cross[spremenljivke]`.

Primer:

Poiščite vektorski produkt vektorjev $\vec{\mathbf{u}} = \{1, 2, 3\}$ in $\vec{\mathbf{v}} = \{3, 4, 2\}$!

```
In:=  u = {1, 2, 3};
      v = {3, 4, 2};
      Cross[u,v]
Out=  {-8, 7, -2}
```

Ukaza `Sum` in `Product`

Ukaz za računanje vsote `Sum` ima sledečo sintakso:
`Sum[funkcija,{i,spodnja_meja,zgornja_meja,korak}]`.

Primer:

Izračunajte vsoto prvih dvanajstih naravnih lihih števil: $\sum_{i=1}^{12}(2i-1)$!

```
In:=  Sum[2i-1,{i,1,12,1}]
Out=  144
```

Ukaz `Product`

Sintaksa ukaza za računanje produkta `Product` se glasi:
`Product[funkcija,{i,spodnja_meja,zgornja_meja,korak}]`.

Primer:

Izračunajte produkt funkcij $i+x$, kjer i preteče vrednosti od 1 do 9 in se povečuje s korakom 2: $\prod_{\text{korak}=2}^{9} i+x$!

```
In:=  Product[i+x,{i,1,9,2}]
Out=  (1 + x) (3 + x) (5 + x) (7 + x) (9 + x)
```

Ukaza `Simplify` **in** `Factor`

Ukaz `Simplify` služi poenostavljanju ukazov in ima enostavno sintakso:
`Simplify[izraz]`.
Podobno sintakso ima tudi ukaz za faktoriziranje `Factor`:
`Factor[izraz]`.

Primera:

```
In:=  A=Sin[x]^2 + Cos[x]^2
      Simplify[A]
      Simplify[(-1 + x) (1 + x)]
Out=  Sin[x]^2 + Cos[x]^2
      1
      -1+x^2
In:=  Factor[-1+x^2]
Out=  (-1 + x) (1 + x)
```

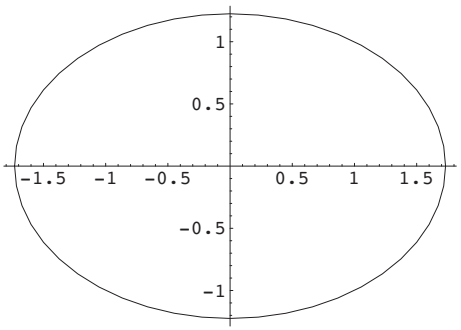
Nalaganje paketov

Večina osnovnih funkcij je dosegljivih takoj po zagonu Mathematice, ne da bi morali za njihovo uporabo najprej naložiti kakšen paket. Obstajajo pa tudi funkcije, ki so del paketa, katerega moramo najprej naložiti, če želimo uporabiti želeno funkcijo. Oglejmo si nalaganje paketov na spodnjem primeru.

Primer:

Če želimo uporabiti funkcijo `ImplicitPlot`, ki omogoča risanje implicitno podanih funkcij, moramo naložiti paket `Graphics`ImplicitPlot`!` Paket naložimo tako, da pred njegovim imenom zapišemo niz `<<`.

```
In:=  <<Graphics`ImplicitPlot`
      ImplicitPlot[x^2 + 2 y^2 == 3, {x, -2, 2}]
```



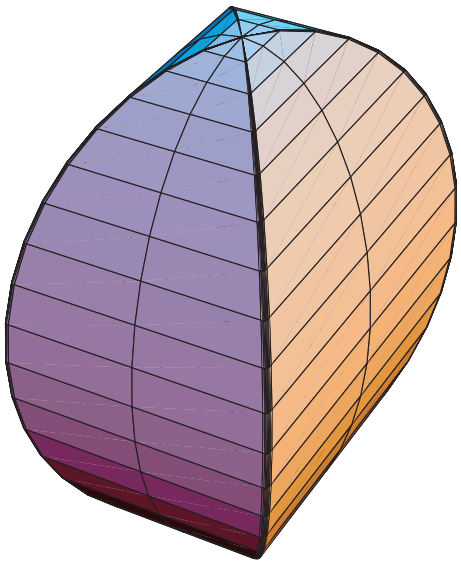

```
Out=
```

9.1.4 Primeri uporabe Mathematice

Ogledali smo si le osnove dela z Mathematico, ki sicer ponuja še mnogo več. Naš cilj je bil osvojiti koncept delovanja in uporabe Mathematice. In ta cilj smo dosegli. Za ilustracijo prve trditve pa si oglejmo še tri primere, ki nazorno demonstrirajo moč Mathematice (zadnja dva primera ilustrirata postopek programiranja v Mathematici):

- Kako narišemo superkvadrik [2], ki je posplošitev elipsoida? Oglejmo si klic funkcije iz paketa `Graphics`ParametricPlot3D`` in njen rezultat, ki je podan na sliki 9.2:

```
In:=  <<Graphics`ParametricPlot3D`
      a1 = 20
      a2 = 20
      a3 = 40
      e1 = 1
      e2 = 0.02
      n = 20
      ParametricPlot3D[
      {a1 Sign[Cos[u] Cos[v]]
      Abs[Cos[u]]^e1 Abs[Cos[v]]^e2,
      a2 Sign[Cos[u] Sin[v]]
      Abs[Cos[u]]^e1 Abs[Sin[v]]^e2,
      a3 Sign[Sin[u]] Abs[Sin[u]]^e1},
      {u, -Pi/2, Pi/2, Pi/n}, {v, -Pi, Pi, Pi/n},
      Lighting->True, Boxed->False]
```



Slika 9.2: Rezultat kode za risanje superkvadrika

- Kako definiramo funkcijo, ki je različna na posameznih intervalih? Najprej podajmo funkcijo, ki jo želimo definirati, nato pa ustrezno kodo te funkcije v Mathematici[4]:

[4] $B(u)$ predstavlja bazno funkcijo pri postopku interpolacije z B-zlepkom.

$$B(u) = \begin{cases} \frac{(2+u)^3}{6}, & -2 \le u \le -1 \\ \frac{(4-6u^2-3u^3)}{6}, & -1 \le u \le 0 \\ \frac{(4-6u^2+3u^3)}{6}, & 0 \le u \le 1 \\ \frac{(2-u)^3}{6}, & 1 \le u \le 2 \\ 0, & \text{sicer} \end{cases}$$

```
In:=  B1[u_]:=(2+u)^3/6;
      B2[u_]:=(4-6 u^2-3 u^3)/6;
      B3[u_]:=(4-6 u^2+3 u^3)/6;
      B4[u_]:=(2-u)^3/6;
      B5[u_]:=0*u;

      B[u_]:=B1[u]/;-2<=u<=-1;
      B[u_]:=B2[u]/;-1<=u<=0;
      B[u_]:=B3[u]/;0<=u<=1;
      B[u_]:=B4[u]/;1<=u<=2;
      B[u_]:=B5[u]/;u>2 || u<-2
```

- Kako avtomatsko ustvarimo matriko velikosti 10×10? Oglejmo si izgled želene matrike in pripadajočo kodo v Mathematici, ki takšno matriko ustvari:

$$\mathbf{M} = \begin{bmatrix} 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 4 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 4 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 4 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 4 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 \end{bmatrix}$$

```
In:=  Num=10
      M=Table[0,{i,1,Num},{i,1,Num}]
      For[i=1,i<Num+1,i++,
        For[j=1,j<Num+1,j++,
          If[i == j,M[[i,j]]=4,
            If[i==j+1 || i==j-1,M[[i,j]]=1, ]
          ]
        ]
      ]
```

V zadnjem primeru smo uporabili tudi vgrajeno funkcijo `Table`, ki je pri resnem delu z Mathematico nepogrešljiva.

Omenimo pa še dve lastnosti Mathematice, ki olajšata delo s programom:

- Mathematica vsebuje odlično pomoč, ki je sicer malo drugače strukturirana, kot smo navajeni, vendar se koncepta hitro privadimo. Pomoč je dosegljiva preko menija "Help" in opcije "Help Browser...".
- Osnovne operacije lahko vnašamo v beležnico tudi s pomočjo klikanja na ustrezne simbole operacij. Orodni vrstici, ki nam to omogočata, najdemo pod menijem "File", pod opcijo "Palettes" in znotraj nje pod opcijo "BasicInput" oziroma "BasicCalculations".

9.1.5 Naloge

1. Kako bi opisali razliko med programoma Mathematica in Matlab?
2. Razložite razliko med simboli za enačenje v Mathematici: `=`, `:=` in `==`!
3. Kako bi v Mathematici poiskali funkcije, ki v svojem imenu vsebujejo besedico `Plot`? Kako pa bi si osvežili spomin o sintaksi ukaza za reševanje sistemov enačb?
4. Omenili smo, da je grafični uporabniški vmesnik Mathematice sestavljen iz dveh delov. Katerih? Opišite značilnosti spodnjega dela vmesnika!
5. Kakšna je relacija med ukazoma `Simplify` in `Factor`? Podajte primer uporabe za vsak ukaz!
6. Kaj izpiše program Mathematica za naslednje vhode?

   a)
   ```
   In:= a=2
        +6
        b=a/2
   ```

   b)
   ```
   In:= Sin[Pi/2]
   ```

   c)
   ```
   In:= D[3x^3+5x^2+x+7,{x,2}]
   ```

7. Narišite parametrični funkciji $\{\frac{1}{2}\cos(t), \sin(t)\}$ in $\{\cos(t), \frac{1}{2}\sin(t)\}$ na intervalu od 0 do 2π na enem grafu. Zagotovite enakomeno razdelitev osi x in y na enote. Na koncu shranite sliko v format EPS.
8. Rešite sistem enačb z dvema neznankama x in y: $y = x^2 + A$, $x = By + 1$. Na koncu shranite ustvarjeno datoteko s končnico `nb` kot dokument urejevalnika LaTeX.

9. Izračunajte:

   a) integral izraza $y = x\sin(x)$,

   b) limito izračunanega integrala, ko gre x proti 0.

10. Dana je matrika:

$$\begin{bmatrix} 20 & 8 \\ 13 & 5 \end{bmatrix}$$

    Izračunajte:

    a) determinanto dane matrike,

    b) inverz dane matrike,

    c) tretjo potenco transponirane matrike.

9.2 Matlab

Matlab je programski paket, namenjen reševanju matematičnih problemov. Za razliko od Mathematice, ki je orientirana v analitično reševanje, pa je Matlab namenjen predvsem numeričnemu načinu reševanja problemov. Je interaktiven sistem, pri katerem uporabnik vtipka ukaz ali zaporedje ukazov, sistem pa jih izvrši in posreduje odgovor. Če nam velika zbirka Matlabovih funkcij ne zadostuje, lahko sami enostavno sprogramiramo dodatne funkcije, ki jih lahko Matlab izvaja enako kot že vgrajene. Programski jezik, ki je vgrajen v Matlab, ima nekaj posebnosti, vendar podpira značilnosti višjih programskih jezikov (Basic, Pascal, C), kot so spremenljivke, kontrolni stavki, klici funkcij ipd. Matlab je namenjen izvajanju matričnih operacij, saj ima vgrajenih veliko funkcij za delo z vektorji in matrikami. Vsebuje tudi dobro podporo za risanje grafov, delo z datotekami in zvokom.

Matlab je plačljiv program, katerega različice so na voljo za več popularnih plaform (Linux, MS Windows). Poleg Matlaba pa lahko dokupimo še številne pakete (imenovane Toolbox), ki so primerni za obdelavo določenih področij. Med te dodatne pakete spadajo recimo paket za statistične obdelave, paket za obdelavo signalov, paket za obdelavo slik, paket za simbolno računanje in podobni. Obstaja pa tudi odprtokodna različica Matlaba, imenovana Octave. Octave je za razliko od Matlaba brezplačen in ga je mogoče enostavno sneti s svetovnega spleta (glej razdelek 9.3), vendar podpira enak jezik in enako sintakso kot Matlab, tako da vsi primeri iz tega poglavja delujejo tudi z Octave-om. Razlog, zakaj kupiti Matlab, je predvsem boljša dokumentacija, večja

podpora in veliko število dodatnih paketov. Če pa ne želimo plačati precej visoke cene za Matlab, pa je Octave dovolj dostojno nadomestilo[5].

9.2.1 Osnovne operacije

Zagon Matlaba

Matlab zaženemo iz ukazne vrstice z ukazom[6]

```
matlab
```

nakar se pojavi Matlabova ukazna vrstica

```
>>
```

V primerih, ki bodo podani v tem razdelku, bodo ukazi, ki jih podajamo Matlabu, vedno zapisani v vrstici, ki se bo začela z ">>". Ostale vrstice pa bodo vsebovale rezultate, ki jih bomo dobili po izvršitvi vsakega ukaza posebej.

Preproste aritmetične operacije

Matlab lahko sedaj uporabimo kot preprost kalkulator:

```
>> 4 + 3
ans = 7
```

Od Matlaba smo zahtevali, naj sešteje števili 4 in 3, in dobili smo rezultat 7. Če od Matlaba eksplicitno ne zahtevamo, da naj rezultat priredi kaki spremenljivki, ga bo priredil posebni spremenljivki `ans`. Vsoto števil 4 in 3 bi lahko izračunali tudi takole:

```
>> a = 4 + 3
a = 7
```

Sedaj je Matlab vsoto števil 4 in 3 priredil spremenljivki `a`. Matlabu lahko preprečimo izpisovanje rezultata tako, da za ukaz dodamo podpičje (;).

```
>> c = 8 - 10;
```

[5] Koristen zastonjski klon je SciLab, ki ima malenkost drugačno sintakso, vendar pride s pretvorniki.

[6] Na priloženi zgoščeniki je program Octave, ki ga zaženemo z ukazom `octave`. Delo s programom od samega zagona naprej je identično delu z Matlabom!

Zgornji ukaz ne bo izpisal ničesar, vendar pa bo razlika števil 8 in 10 shranjena v spremenljivki `c`. Če sedaj Matlab povprašamo po njeni vrednosti, bomo dobili odgovor:

```
>> c
c = -2
```

Spremenljivke lahko uporabljamo tudi kot argumente operatorjev in funkcij, ne samo kot mesto, kamor shranimo rezultat:

```
>> a = 4
a = 4
>> b = 3
b = 3
>> c = a * b
c = 12
>> d = a / b
d = 1.3333
```

Matlab vsebuje veliko matematičnih funkcij, med njimi na primer inverzni kosinus (`acos`):

```
>> acos(0)
ans = 1.5708
```

Spisek vseh funkcij lahko dobimo s pomočjo ukaza `help`. Če nas zanima kaj izračuna funkcija `sin`, lahko o tem povprašamo Matlab z ukazom `help sin`.

Računanje s kompleksnimi števili

Matlab podpira kompleksna števila, ki jih podamo tako, kot smo navajeni iz matematike:

```
>> a = 1-2i
a = 1 - 2i
>> b = 2+i
b = 2 + 1i
>> a/b
ans = 0 - 1i
```

S kompleksnimi števili lahko v Matlabu računamo tako kot z realnimi. Velja še omeniti, da sta `i` in `j` dejansko spremenljivki, ki imata vrednost imaginarne enote. Če spremenimo vrednost spremenljivki `i`, potem kompleksnih števil ne moremo več podajati tako kot v prejšnjem primeru

(lahko pa še vedno uporabimo spremenljivko `j`, če tudi tej nismo spremenili vrednosti). Iz zagate nas lahko reši ukaz `clear i`, ki zbriše vrednost spremenljivke `i`; tako dobi `i` nazaj vrednost imaginarne enote.

9.2.2 Delo z vektorji in matrikami

Osnovne aritmetične operacije nad matrikami. Matlab podpira delo z vektorji in matrikami, na katerih lahko opravljamo razne operacije. Matrike in vektorje je smiselno označevati z velikimi tiskanimi črkami, da jih lažje ločimo od skalarjev, ki se običajno označujejo z malimi tiskanimi črkami. Dejansko pa Matlab skalarje obravnanava tako kot matrike velikosti 1×1. Kako v Matlabu podajamo vektorje in kako računamo z njimi, si lahko ogledamo na spodnjem primeru:

```
>> [1 2 3]+[2 3 4]
ans =
  3  5  7
```

Vektor podamo tako, da med oglatim oklepajem (`[`) in zaklepajem (`]`) naštejemo elemente vektorja, ki morajo biti ločeni vsaj z enim presledkom ali pa z vejico. V zgornjem primeru smo sešteli dva vektorja s tremi elementi. Matriko z več vrsticami podamo tako, da vrstice med sabo ločimo s podpičjem ali pa s prehodom v novo vrsto. Seveda morajo imeti vse vrstice enako število elementov. Tudi vektorje in matrike lahko priredimo spremenljivkam, kar ilustrira spodnji primer:

```
>> A = [1 2 3]
A =
  1  2  3
>> B = [ 1 2 3; 3 4 5; 5 6 7]
B =
  1  2  3
  3  4  5
  5  6  7
>> C = [ 2 4
> 5 6 ]
C =
  2  4
  5  6
```

Ko smo pri matriki `C` v zgornjem primeru vnesli prvo vrstico, je Matlab zaznal, da naš vnos ni dokončan, zato nas je v naslednji vrstici z znakom `>` obvestil, da moramo vnos dokončati. Šele ko je vnos dokončan, Matlab izvede zahtevano operacijo.

Če nas zanima, kateri element se nahaja v drugi vrstici in tretjem stolpcu matrike `B`, lahko to izvemo takole:

```
>> B(2,3)
ans = 5
```

Vrednost zadnjega desnega elementa pa dobimo z: `>> B(end,end)`.

Pri vektorjih (torej pri matrikah, ki imajo število stolpcev ali število vrstic enako 1), zadostuje, če podamo eno samo število, ki potem določa indeks elementa vektorja:

```
>> A = [ 0 2 3 7];
>> A(4)
ans = 7
>> A(1,4)
ans = 7
```

Operacija množenja (`*`), ki smo jo spoznali v prejšnjem razdelku, ima pri računanju z matrikami dve vlogi. Če je eden izmed argumentov število in drugi matrika, oziroma vektor, bo Matlab izvedel množenje matrike s skalarjem:

```
>> 3 * [1 2 3 4]
ans =
   3   6   9  12
```

Matlab opravi operacije seštevanja, odštevanja, množenja in deljenja tudi, če je en argument matrika drugi pa navaden skalar. Rezultat take operacije je matrika, katere elemente dobimo tako, da operacijo izvršimo na vsakem elementu matrike, ki nastopa kot argument operacije:

```
>> 2 + [1 2 3 4]
ans =
  3  4  5  6
>> [1 2 3 4] / 8
ans =
  0.12500  0.25000  0.37500  0.50000
```

Če sta oba argumenta pri množenju matriki, pa bo Matlab izvedel klasično množenje med matrikami, seveda, če imata argumenta ustrezni dimenziji.

```
>> A = [ 0.5 2 1; 1 0 2; 1 0.5 0.5];
>> B = [ 1; 2; 3];
>> A*B
ans =
```

```
  7.5000
  7.0000
  3.5000
```

Rezultat množenja matrik `A` in `B` je matrika (vektor) s tremi vrsticami in enim samim stolpcem in v taki obliki je Matlab tudi izpisal rezultat. Matlab pozna tudi operacijo `.*`, ki zmnoži istoležne elemente v dveh matrikah enakih dimenzij.

```
>> A = [ 1 0 1; 0 0 2; 2 2 2];
>> B = [ 3 7 2; 2 3 1; 1 1 3];
>> A.*B
ans =
  3  0  2
  0  0  2
  2  2  6
```

Tehnike sestavljanja novih matrik

Matriko ali vektor lahko podamo kot argument običajnim matematičnim funkcijam. Rezultat bo matrika ali vektor, kjer je vsak element enak vrednosti, ki jo dobimo, če bi funkciji kot argument podali istoležni element v matriki, ki jo podamo kot argument:

```
>> X = [1 2 4 8 16 32 64];
>> Y = log2(X)
Y =
  0  1  2  3  4  5  6
```

Še posebej pri eksperimentih pride zelo prav generator naključnih števil. V ta namen ima Matlab funkcijo `rand`, ki nam zgenerira matriko, katere elementi so naključna realna števila med 0 in 1. Če ukazu `rand` podamo en sam parameter n, potem bomo dobili naključno kvadratno matriko velikosti $n \times n$. Če pa mu podamo dva parametra n in m, pa zgenerira naključno matriko velikosti $n \times m$:

```
>> rand(2,3)
ans =
  0.560308  0.115751  0.771474
  0.263282  0.026583  0.138481
```

Sintaksa Matlaba nam omogoča, da kot elemente matrike podamo tudi spremenljivke:

```
>> a = 1;
```

```
>> b = 0;
>> c = 2;
>> D = [ a b c; b 3 c; c 10 a]
D =
   1   0   2
   0   3   2
   2  10   1
```

Lahko pa matriko sestavimo tudi iz manjših podmatrik. Pri tem veljajo podobna pravila kot pri sestavljanju matrik iz samih skalarjev. Notacija `C = [A B]` pomeni, da bi matriko `C` dobili tako, da bi na desno stran matrike `A` zapisali matriko `B`. Seveda morata `A` in `B` imeti enako število vrstic. Notacija `C = [A; B]` pa pomeni, da bi matriko `C` dobili tako, da bi matriko `B` zapisali pod matriko `A`. V tem primeru morata `A` in `B` imeti enako število stolpcev. Tule lahko vidimo primer, kako sestavimo matriko iz nekaj podmatrik:

```
>> A = [ 1 2; 3 4];
>> B = [ 0; 0];
>> C = [ 6 7 8];
>> D = [[ A B; C] [1; 1; 1]]
D =
  1  2  0  1
  3  4  0  1
  6  7  8  1
```

Če hočemo določeno vrednost vpisati v matriko na mesto, ki presega trenutne dimenzije matrike, Matlab matriko samodejno razširi:

```
>> A = [ 1 2; 3 4];
>> A(2,4) = 3
A =
  1  2  0  0
  3  4  0  3
```

Iz matrike lahko "izrežemo" manjšo podmatriko. Prvo vrstico matrike `D` dobimo na naslednji način:

```
>> D = [  1  2  0  1; 3  4  0  1; 6  7  8  1];
>> D(1,:)
ans =
  1  2  0  1
```

Na podoben način bi lahko dobili drugi stolpec matrike `D`:

```
>> D(:,2)
ans =
  2
  4
  7
```

V splošnem lahko iz matrike "izrežemo" katerokoli podmatriko, s tem da navedemo interval vrstic in interval stolpcev, ki naj bodo vključeni v podmatriko.

```
>> D(1:2,1:3)
ans =
  1  2  0
  3  4  0
```

Celoštevilski interval v Matlabu navedemo kot `a:b`, kjer je `a` začetna vrednost na intervalu, `b` pa končna. Če kot interval navedemo samo `a`, to dejansko pomeni `a:a`. Pri določanju podmatrik iz matrike pa lahko za definicijo intervala uporabimo samo dvopičje (`:`), ki dejansko pomeni `1:m`, kjer je `m` število vrstic ali stolpcev v matriki. Če potrebujemo vektor, pri katerem je razlika med $i+1$-im in i-tim elementov vedno enaka, ga lahko generiramo s pomočjo dvopičja:

```
>> A = 1:3:20
A =
  1   4   7  10  13  16  19
```

Pri takem načinu podajanja vektorja pomeni prvo število vrednost prvega elementa v vektorju, drugo število razliko (korak) med sosednimi elementi vektorja in tretje največjo možno vrednost elementa v vektorju. Če je razlika med sosednimi elementi enaka 1, potem lahko drugi parameter izpustimo. Ukaz `1:10` generira vektor, katerega elementi so naravna števila od 1 do 10.

Dimenzije matrike lahko izvemo s pomočjo funkcije `size`:

```
>> A = [1 2 3];
>> size(A)
ans =
  1  3
```

Linearna algebra

Matriko lahko transponiramo, kvadratnim matrikam pa lahko izračunamo inverzno matriko ali determinanto:

```
>> A = [2 2; 4 1; 1 0]
A =
  2  2
  4  1
  1  0
>> A'
ans =
  2  4  1
  2  1  0
>> B = [ 1 2; 2 1];
>> inv(B)
ans =
  -0.33333   0.66667
   0.66667  -0.33333
>> det(B)
ans = -3
```

Problem, ki ga velikokrat srečamo v matematiki ter drugih področjih, je reševanje sistema n linearnih enačb z n neznankami. Tega lahko z Matlabom zelo elegantno rešimo. Imamo na primer naslednji sistem enačb:

$$\begin{aligned} 2x + 3y &= 11 \\ 4x - y &= 1 \end{aligned} \tag{9.1}$$

Nalogo rešimo tako, da v določeno matriko damo koeficiente pri neznankah na levi strani enačb, v določen vektor pa koeficiente na desni strani enačb in uporabimo matrični operator \.

```
>> B = [2 3; 4 -1];
>> A = [11; 1];
>> B\A
ans =
  1
  3
```

Zelo enostavno lahko poiščemo lastne vrednosti matrik:

```
>> B = [ 2 1; 1 2];
>> eig(B)
ans =
  1
  3
```

Vgrajenih funkcij za delo z matrikami je v Matlabu še precej, njihove specifikacije pa lahko preberemo v kakšnem posebnem priročniku ali pa s pomočjo vgrajene pomoči z ukazom `help`.

9.2.3 Delo s polinomi

Sestavljanje polinomov

Polinome v Matlabu predstavimo kar z vektorjem, ki predstavlja njihove koeficiente. Polinom $p(x) = 3x^3 - 7x^2 + 2x - 3$ definiramo torej takole:

```
>> P = [3 -7 2 -3];
```

Vrednost polinoma, predstavljenega z vektorjem `P`, v posamezni točki lahko sedaj izračunamo s pomočjo funkcije `polyval`:

```
>> polyval(P,1)
ans = -5
```

Polinome lahko v Matlabu med sabo množimo in delimo s pomočjo funkcij `conv` in `deconv`:

```
>> P1 = [ 1 2 -1 1];
>> P2 = [ 1 -1];
>> conv(P1,P2)
ans =
   1   1  -3   2  -1
>> deconv(P1,P2)
ans =
  1  3  2
```

Analiza polinomov

Enostavno lahko izračunamo tudi odvod polinoma:

```
>> P = [3 -7 2 -3];
>> polyder(P)
ans =
    9  -14    2
```

Tudi izračun drugega, tretjega in višjih odvodov ne predstavlja posebnih težav:

```
>> polyder(polyder(P))
ans =
   18  -14
```

Polinomu poiščemo ničle z ukazom `roots`:

```
>> roots(P)
ans =
  2.23523 + 0.00000i
  0.04905 + 0.66706i
  0.04905 - 0.66706i
```

Funkcija `poly` je inverzna funkciji `roots`. Če imamo v nekem vektorju podane ničle polinoma, zna `poly` iz njih izračunati koeficiente ustreznega polinoma:

```
>> X = [0 1 2];
>> poly(X)
ans =
   1  -3   2   0
```

Matlabov odgovor nam pove, da je polinom $x^3 - 3x^2 + 2x$ tisti, ki ima ničle v točkah 0, 1 in 2. Matlab pa zna izračunati tudi koeficiente polinoma, ki se najbolj prilega množici točk. Podati mu moramo vektor x koordinat točk, vektor y koordinat točk ter stopnjo polinoma. Recimo da želimo izvedeti, kakšno enačbo ima premica, ki se najbolj prilega točkam $(0, 1), (2.5, 3), (3, 3.75)$ in $(4, 5)$. Problem lahko rešimo na naslednji način:

```
>> X = [0 2.5 3 4];
>> Y = [1 3 3.75 5];
>> polyfit(X,Y,1)
ans =
  0.97482
  0.87230
```

Enačba premice, ki je najbližje točkam, je torej $y = 0.97482x + 0.87230$.

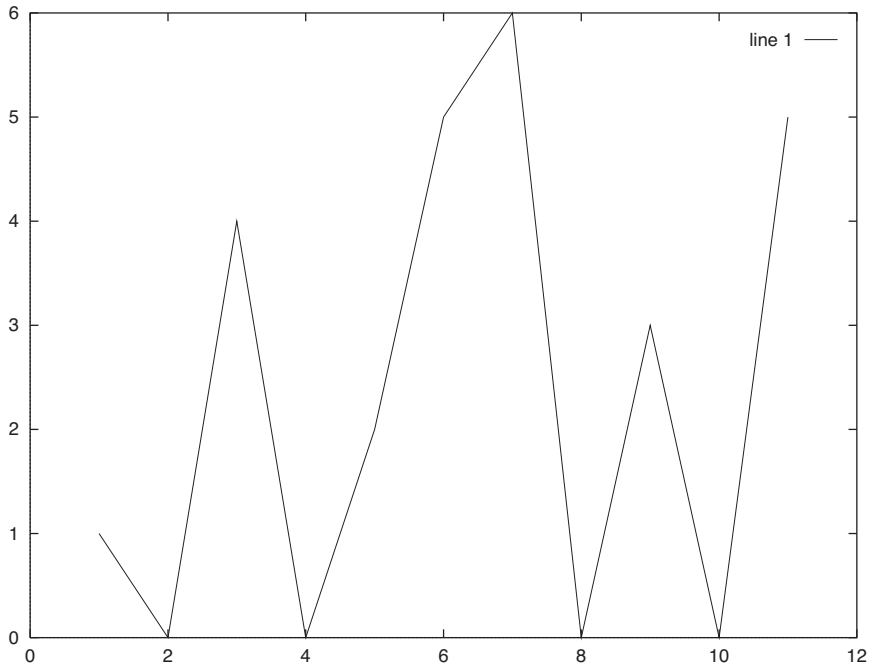
9.2.4 Grafika

Izrisovanje funkcij ene spremenljivke

Matlab ima dobro podprto delo z grafiko, predvsem risanje najrazličnejših grafikonov. Če imamo v nekem vektorju vrednosti, jih lahko narišemo s pomočjo ukaza `plot`:

```
>> Y = [ 1 0 4 0 2 5 6 0 3 0 5 ];
>> plot(Y);
```

Izvedba zgornjih ukazov nam izriše graf, prikazan na sliki 9.3. Vektor vrednosti se nariše tako, da je njegov prvi element vrednost na osi y v točki $x = 1$, drugi element vrednost na osi y v točki $x = 2$ itd. Funkciji



Slika 9.3: Enostaven prikaz vektorja vrednosti

`plot` lahko podamo dva vektorja enake dolžine, pri katerem so elementi drugega vektorja vrednosti na osi y, kjer ima koordinata x vrednost istoležnega elementa prvega vektorja.

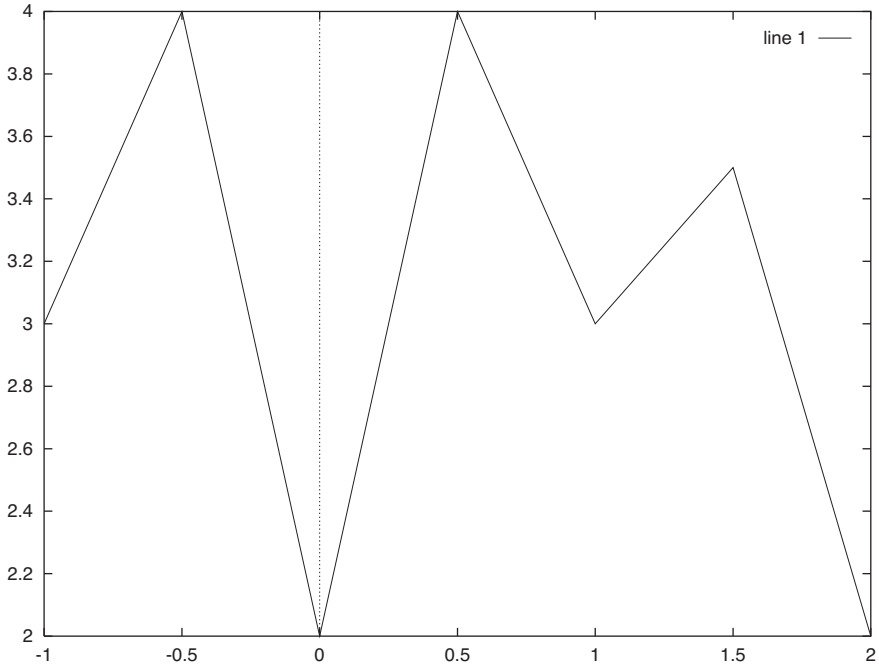
```
>> X = [-1 -0.5 0 0.5 1 1.5 2 ];
>> Y = [3 4 2 4 3 3.5 2];
>> plot(X,Y);
```

To zaporedje ukazov nam izriše grafikon na sliki 9.4. Ukazu `plot` lahko kot zadnji parameter podamo format grafikona. Tako lahko na primer narišemo stopničast grafikon, kot je na sliki 9.5.

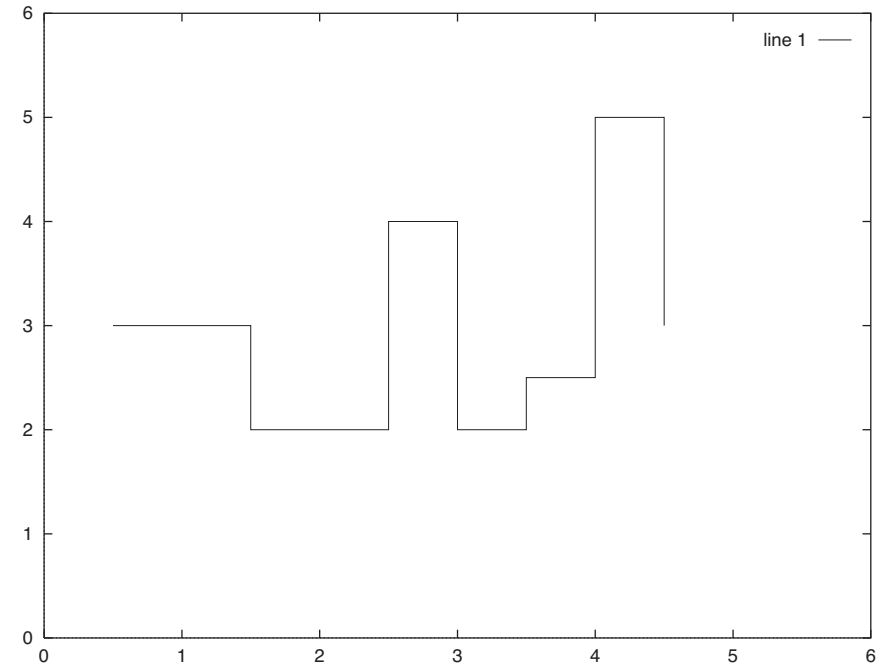
```
>> X = [0.5 1.5 2.5 3 3.5 4 4.5];
>> Y = [ 3   2   4  2 2.5 5 3];
>> axis([0 6 0 6]);
>> plot(X,Y,"L");
```

Vmes smo uporabili še funkcijo `axis`, s katero definiramo območje grafa, ki ga hočemo prikazati. Za ostale opcije ukaza `plot` lahko vprašamo kar Matlab s `help plot`. Podobno lahko naredimo tudi s funkcijo `bar`, ki vrednosti v vektorju predstavi z različno visokimi stolpci:

```
>> bar(X,Y);
```



Slika 9.4: Enostaven prikaz vektorja vrednosti pri različnih vrednostih na osi x



Slika 9.5: Eden izmed načinov prikaza grafikonov

Izrisovanje funkcij dveh spremenljivk

Z Matlabom lahko narišemo tudi ploskve v treh dimenzijah. Za to potrebujemo funkcijo `mesh`, ki ji lahko kot argument podamo samo matriko, katere elementi predstavljajo koordinate z točk na ploskvi. Če želimo, lahko z dvema vektorjema podamo tudi vrednosti na koordinatnih oseh x in y. Sedaj lahko, na primer, narišemo funkcijo $\log_2(xy)$ na območju $[1 \ldots 10] \times [1 \ldots 10]$:

```
>> X = 1:10;
>> Y = 1:10;
>> Z = log2(X'*Y);
>> mesh(X,Y,Z);
```

Ploskev, ki jo izriše ukaz `mesh` iz prejšnjega primera, je prikazana na sliki 9.6. Če ukazu `mesh` kot parametre posredujemo tri matrike enakih dimenzij, nam ta nariše povezano mrežo točk, kjer prva matrika predstavlja x koordinate, druga y koodrinate in tretja z koordinate.

Lep primer 3D ploskve pa lahko dobimo s pomočjo priložene funkcije `sombrero` (slika 9.7):

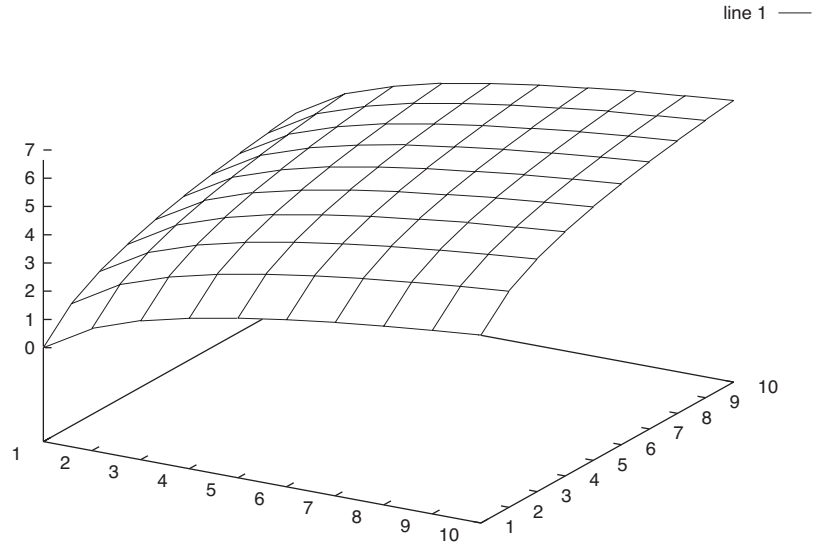
```
>> sombrero(50);
```

Prikaz slik

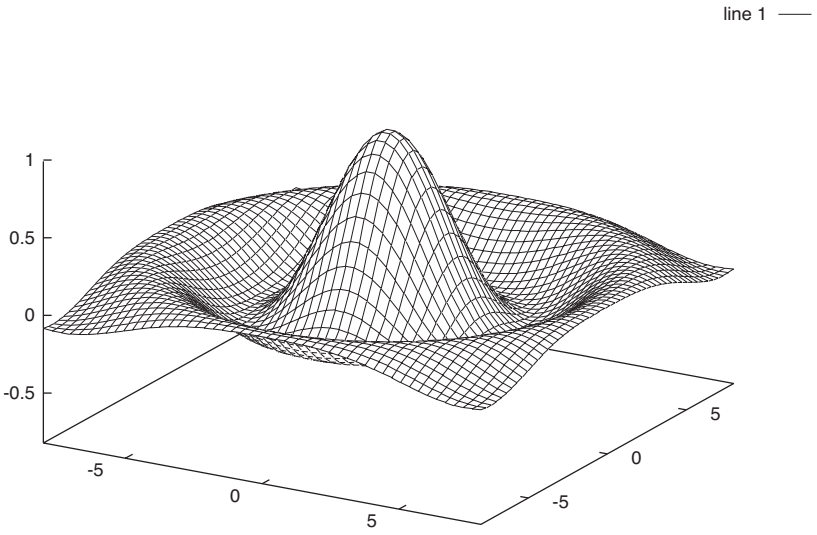
Matriko lahko s pomočjo funkcije `image` narišemo kot sliko, pri čemer vrednost vsakega elementa predstavlja indeks barve v paleti. Paleta je definirana kot matrika velikosti $m \times 3$, pri čemer so vsi elementi med 0 in 1. Elementi vsake vrstice predstavljajo intenziteto rdeče, zelene in modre komponente v barvi, ki jo ta vrstica predstavlja. Ta paleta ima na začetku $m = 64$ barv, ki pa so le različni nivoji sivine. Seveda lahko definiramo tudi svojo paleto. Spodnji primer kaže, kako lahko isto matriko narišemo z različnimi barvami:

```
>> A = 1:50;
>> B = A';
>> S = 64*abs(sin(log2(B*A)));
>> image(S);
>> P = rand(64,3);
>> colormap(P);
>> image(S);
```

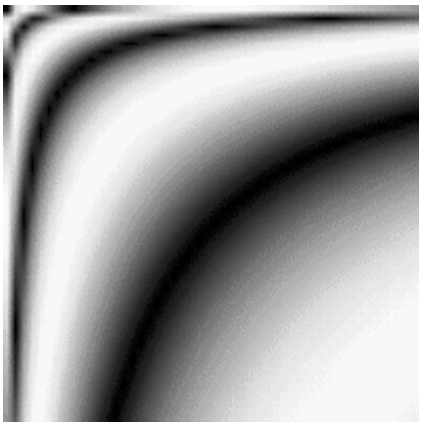
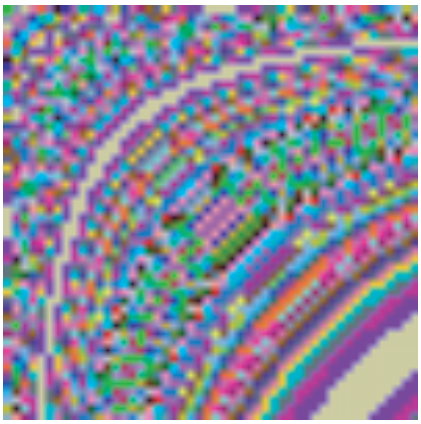
Ko izrišemo matriko z začetno paleto, dobimo izris, prikazan na sliki 9.8(a), ko pa postavimo v paleto 64 naključnih barv, pa dobimo izris na sliki 9.8(b).



Slika 9.6: Primer 3D ploskve



Slika 9.7: Sombrero


(a)

(b)

Slika 9.8: (a) Matrika, izrisana z originalno paleto barv, (b) matrika, izrisana z naključno paleto barv

Funkcija `meshgrid` nam lahko en vektor razširi v kvadratno matriko, v kateri je vsaka vrstica enaka prav temu vektorju:

```
>> X = [0 1 2 3];
>> meshgrid(X)
ans =
  0  1  2  3
  0  1  2  3
  0  1  2  3
  0  1  2  3
```

Če pa funkciji `meshgrid` podamo kot argumente dva vektorja (naj bosta dolžin n in m), dobimo kot rezultat dve matriki (obe sta velikosti $n \times m$), od katerih prva vsebuje vrstice, ki so enake prvemu vektorju, druga pa stolpce, ki so enaki drugemu vektorju:

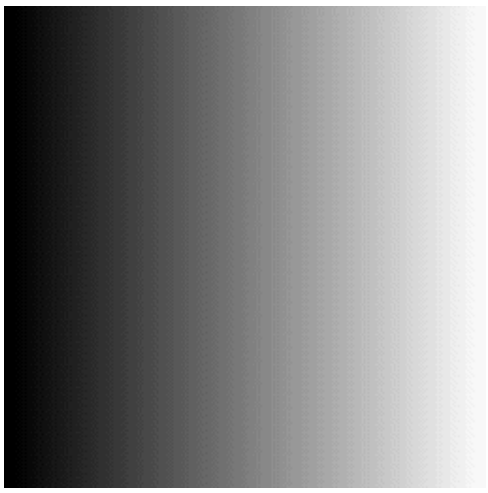
```
>> X = 1:4;
>> Y = 5:7;
>> [A B] = meshgrid(X,Y)
A =
  1  2  3  4
  1  2  3  4
  1  2  3  4
B =
  5  5  5  5
  6  6  6  6
  7  7  7  7
```

Vidimo tudi, da lahko funkcije v Matlabu vračajo rezultat, ki je sestavljen iz večih komponent.

S pomočjo funkcije `meshgrid` lahko zelo enostavno izrišemo paleto barv:

```
>> S = size(colormap)
S =
  64   3
>> X = 1:S(1);
>> image(meshgrid(X));
```

Na sliki 9.9 vidimo, kakšno paleto barv ima Matlab, potem ko ga zaženemo.



Slika 9.9: Začetna paleta barv v Matlabu

9.2.5 Programiranje v Matlabu

Matlab vsebuje preprost, a dovolj močan višjenivojski programski jezik, s katerim si lahko še bolj olajšamo delo in avtomatiziramo postopke. Hkrati pa lahko Matlab razširimo z novimi funkcijami, ki jih sami napišemo in dodamo v pot, kjer jih potem ob vsakem klicu tudi poišče. Programiramo lahko interaktivno, to je v Matlabovi lupini, kjer se ukazi izvajajo oziroma zaporedje ukazov izvede, takrat ko jih zaključimo (to je ponavadi s pritiskom na ENTER, ni pa nujno) ali pa tako, da funkcijo ali program napišemo v poljubnem urejevalniku besedila.

Spremenljivke v Matlabu lahko vsebujejo števila (cela, realna ali kompleksna), matrike ter nize znakov. Kot smo že videli v primerih, spremenljivkam ni treba vnaprej definirati tipov niti dimenzij, tako kot pri nekaterih drugih višjih programskih jezikih. Prav tako lahko spremenljivki, ki trenutno vsebuje matriko, priredimo niz znakov in podobno. Matlab loči velike in male črke, torej črka `a` ni enaka črki `A`.

Imena spremenljivk in tudi funkcij so v Matlabu lahko sestavljena iz začetne črke, ki ji sledi poljubna kombinacija črk, številk in posebnih znakov. Dolžina imen je omejena na 31 znakov.

Nadzor nad tokom izvajanja programa se izvaja s pomočjo ukazov `if`, `else`, `elseif`, `for`, `while`, `end`, `break` in `return`. Ukazi imajo podoben učinek kot isti ukazi v programskih jezikih Pascal in C, zato si njihovo delovanje poglejmo kar na primerih.

Programiranje v Matlabovi lupini

Ukaz `if` nam omogoča pogojno izvajanje ukazov. Uporabimo ga torej v primeru, če želimo zagotoviti, da se bodo vsi ukazi, ki so navedeni do ustreznega ukaza `end`, `else` ali `elseif`, izvedli samo, če je izpolnjen pogoj, ki je naveden za ukazom `if`.

```
>> a = 3
a = 3
>> if a < 4
> printf("a je manjsi od 4\n");
> end
a je manjsi od 4
>> if a < 3
> printf("a je manjsi od 3\n");
> end
```

V zgornjem primeru smo mimogrede uporabili še funkcijo `printf`, ki dela enako kot v jeziku C, torej izpisuje na zaslon. V prvem primeru, ko je pri ukazu `if` nastopal pogoj `a < 4`, je bil le-ta izpolnjen, zato se je stavek "a je manjsi od 4" tudi dejansko izpisal. Drugi pogoj `a < 3` ni bil izpolnjen, zato se stavek "a je manjsi od 3" ni izpisal na terminalu. Ko v Matlabovi lupini napišemo ukaz `if a < 3`, Matlab tega ukaza ne izvrši takoj, ampak čaka, da ga dokončamo. Svoje čakanje na dokončanje ukaza nam ponazori z znakom `>` na začetku vnosne vrstice. Ukaz `if` se mora tako kot ukaza `for` in `while` končati z ukazom `end`. Lahko pa s pomočjo ukazov `if`, `else` in `elseif` preverimo več pogojev, kar je razvidno iz naslednjega primera:

```
>> a = 3
a = 3
>> if a == 1
> printf("a je enak 1\n");
> elseif a == 2
> printf("a je enak 2\n");
> elseif a == 3
> printf("a je enak 3\n");
```

```
> else
> printf("a je vecji kot 3\n");
> end
a je enak 3
```

Če pogoj, ki sledi besedici `if`, ni izpolnjen, se izvršijo ukazi, ki sledijo prvemu ukazu `elseif`, pri katerem je pogoj izpolnjen, če seveda kakšen ukaz `elseif` sploh obstaja. Če pri nobenem izmed ukazov pogoj `elseif` ni izpolnjen, se izvršijo ukazi, ki sledijo ukazu `else`, če seveda ta obstaja.

Ukaz `while` nam pride prav, kadar želimo nekaj ukazov ponavljati, dokler je določen pogoj izpolnjen. Tudi ukaz `while` mora biti zaključen z ukazom `end`, zato da Matlab ve, katere ukaze mora izvajati:

```
>> n = 0;
>> sum = 0;
>> while n <= 10
> sum = sum + n;
> n = n + 1;
> end
>> sum
sum = 55
```

V zgornjem primeru smo izračunali vsoto naravnih števil od 1 do 10.

Eden najbolj uporabnih ukazov v Matlabu je vsekakor `for`. Ta nam omogoča, da se vsi ukazi do ustreznega ukaza `end` izvršijo tolikokrat, kot je število elementov v zančni matriki, ki jo podamo ob ukazu `for`. V prvi iteraciji ima zančna spremenljivka vrednost prvega elementa v zančni matriki, ob vsaki naslednji iteraciji pa dobi vrednost naslednjega elementa v zančni matriki. Uporabnost ukaza `for` ilustrira spodnji primer:

```
>> sum = 0
sum = 0
>> for n = 1:10
> sum = sum + n;
> end
>> sum
sum = 55
>> A = [ 3 8 4 0 2 5 ];
>> for n = A
> printf("%d \n",n);
> end
3
8
4
0
```

```
2
5
```

V prvi zanki `for` smo zopet izračunali vsoto naravnih števil od 1 do 10. Vlogo zančne spremenljivke igra spremenljivka `n`, vlogo zančne matrike pa `1:10`, ki nam dejansko definira vektor naravnih števil od 1 do 10, kot smo videli v razdelku 9.2.2. V drugi zanki `for` pa smo izpisali vse elemente v vektorju `A`.

Ukaz `break` se obnaša popolnoma enako kot v C-ju, kar pomeni, da se uporablja za takojšen izhod iz zank `while` ali `for`.

Ukaz `return` je podoben ukazu `break`, le da povzroči, da se izvajanje v neki funkciji konča.

Vmes smo spoznali še funkcijo `printf`, ki je namenjena izpisovanju na zaslon. Funkcij, ki so uporabnikom na razpolago, je v Matlabu še veliko. Več o njih lahko izvemo s pomočjo ukaza `help` ali pa iz kakšnega priročnika za Matlab.

Pisanje novih funkcij in programov

Sedaj imamo dovolj podlage, da se naučimo še pisanja svojih lastnih funkcij za Matlab. Za preprost zgled lahko vzamemo funkcijo fakultete, ki izračuna produkt prvih n števil. V poljubnem urejevalniku besedil napišemo spodnje besedilo, ki ga shranimo na disk pod imenom *fakulteta.m*:

```
function f = fakulteta(n)
% fakulteta(n) - izracuna fakulteto stevila n
% fakulteta je definirana kot produkt prvih n naravnih stevil

f = 1;
for k=2:n
   f = f * k;
end

return;
```

V prvi vrstici smo z ukazom `function` Matlabu povedali, da gre tu za definicijo funkcije. Dejansko lahko v Matlabu na podoben način napišemo tudi celoten program, ki ga potem zaženemo iz Matlabove vnosne vrstice, ki pa ne more imeti parametrov, ker ni funkcija. Program v Matlabu se od funkcije dejansko razlikuje samo po tem, da ne vsebuje ukaza `function`. Preostali del prve vrstice v zgornjem primeru pove, da bo vrednost, ki jo bo funkcija vrnila, shranjena v spremenljivki `f`, `fakulteta` bo ime funkcije (to ime se mora ujemati z imenom datoteke, v kateri je

funkcija shranjena) in `n` bo edini parameter te funkcije. Druga in tretja vrstica se začneta z znakom %, ki označuje, da je besedilo do konca vrstice komentar, in ga Matlab preskoči. V funkcijah je koristno imeti take komentarje, ker se izpišejo, če izvedemo ukaz `help fakulteta` iz vnosne vrstice. Če pišemo funkcije za kak večji projekt, pa je pisanje takih komentarjev skorajda obvezno, saj hkrati rešuje problema dokumentiranosti kode in takojšnje pomoči v programu. Komentarji, ki so navedeni takoj za deklaracijo funkcije (to je vrstica, ki vsebuje ukaz `function`), igrajo torej vlogo pomoči v Matlabu in so za uporabnike večkrat zelo koristni.

V naslednjih vrsticah smo dejansko izračunali fakulteto števila n in rezultat shranili v spremenljivko `f`. Izvajanje funkcije se konča takrat, ko Matlab naleti na ukaz `return` ali pa na konec datoteke. V zgornjem primeru je ukaz `return` torej nepotreben, uporaben pa je, če želimo izvajanje funkcije končati nekje na sredi vseh ukazov v definiciji funkcije. Vse spremenljivke, ki jih uporabimo v funkciji, so lokalne, kar pomeni, da če v funkciji določeni spremenljivki priredimo določeno vrednost in v Matlabovi ukazni vrstici vprašamo po vrednosti iste spremenljivke, ne bomo nujno dobili iste vrednosti. Velja tudi obratno, če v Matlabovi ukazni vrstici priredimo vrednost določeni spremenljivki, le-ta ne bo nujno imela iste vrednosti v funkciji. Če želimo imeti na primer spremenljivko `A`, za katero bi želeli, da lahko njeno vrednost uporabljamo v več funkcijah, moramo v vsaki izmed teh funkcij uporabiti ukaz `global A`.

Ko definicijo funkcije shranimo v datoteko, jo lahko enostavno preizkusimo v Matlabu:

```
>> fakulteta(4)
ans = 24
>> fakulteta(5)
ans = 120
>> fakulteta(6)
ans = 720
```

Funkcije v Matlabu lahko vrnejo tudi več rezultatov naenkrat. Vsakega izmed njih predstavlja po ena spremenljivka. Tukaj podajmo kodo funkcije `deli` (shranimo jo v datoteko *deli.m*), ki deli dve števili in kot rezultat vrne rezultat celoštevilskega deljenja in ostanek

```
function [r o] = deli(a, b)
% function [r o] = deli(a b) - celostevilsko deli stevilo a
% s stevilom b in vrne rezultat ter ostanek.

r = floor(a/b);
o = a - b * r;
```

Rezultat, ki ga vrača ta funkcija, smo podali kot vektor z dvema komponentama, ki bosta vsebovali rezultat in ostanek pri celoštevilskem deljenju. Če nas zanima samo rezultat, bomo funkcijo poklicali na običajen način:

```
>> deli(17,6)
ans = 2
```

V tem primeru Matlab vrne kot rezultat samo prvo komponento rezultata. Če pa nas zanima tudi ostanek pri deljenju, moramo to Matlabu posebej povedati:

```
>> [rez ost] = deli(17,6)
rez = 2
ost = 5
```

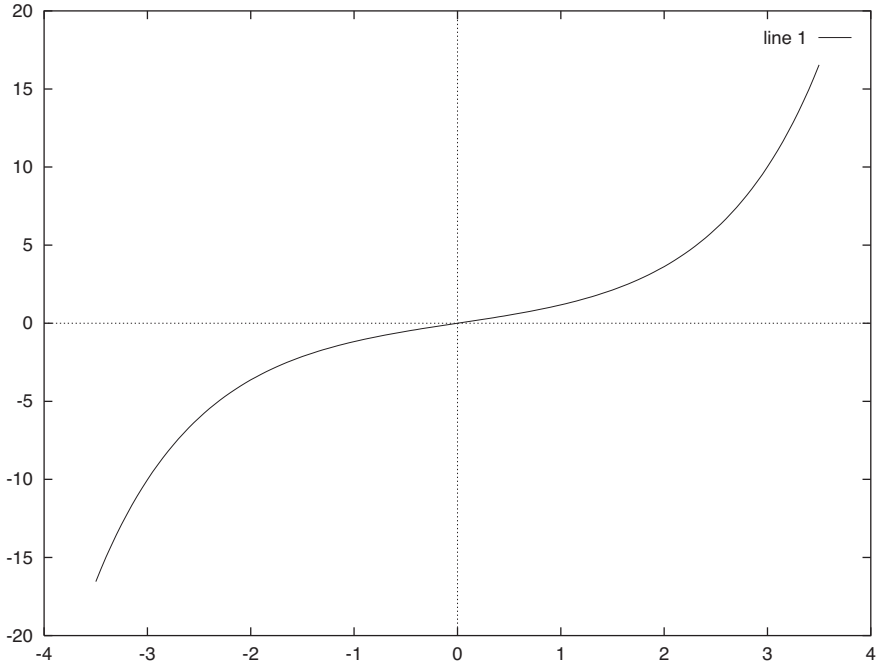
Za konec pa si poglejmo še primer malo bolj zapletene funkcije, ki nam izriše graf poljubne funkcije:

```
function graf_funkcije(f, xmin, xmax, dx)
% graf_funkcije(f, xmin, xmax, dx) - izrise graf funkcije f
% od tocke x = xmin do tocke x = xmax z razmikom dx med tockami

X = xmin:dx:xmax;
u = strcat('Y=',f,'(X);');
eval(u);
plot(X,Y);
```

Ker ta funkcija ne bo vračala nobenega rezultata, ni potrebno specificirati spremenljivke, v kateri bo rezultat. Najprej smo spremenljivki X priredili matriko, v kateri so števila med `xmin` in `xmax` z razmikom `dx`. Nato smo s pomočjo funkcije `strcat`, ki spoji dva ali več nizov znakov skupaj, generirali spremenljivko `u`. Če je v spremenljivki `func` niz `cos`, bo imela spremenljivka `u` vrednost "`Y=cos(X);`", kar je natanko tisti ukaz, ki ga moramo izvršiti, da se v matriko `Y` izračunajo kosinusi elementov matrike `X`. S funkcijo `eval` izvršimo ukaz, ki se kot niz znakov nahaja zapisan v spremenljivki `u`. Na koncu pa s pomočjo funkcije `plot` narišemo graf funkcije. Preprosto in učinkovito torej. Je pa res, da ima ta funkcija nekaj omejitev. Za ime funkcije, ki jo želimo narisati, moramo podati enostavno funkcijo in ne sestavljene, poleg tega pa se mora znati izvajati tudi, če ji kot argument podamo matriko. Vendar bi se z malo truda tudi te težave dalo rešiti. Preostane nam samo še to, da si pogledamo primer uporabe novo definirane funkcije `graf_funkcije`, ki jo moramo shraniti v datoteko *graf_funkcije.m*:

```
>> graf_funkcije('sinh',-3.5,3.5,0.05)
```

Z zgornjim ukazom smo narisali graf funkcije hiperbolični sinus v mejah od $x = -3.5$ do $x = 3.5$ z razmikom 0.05 med točkami podanega intervala. Rezultat lahko vidimo na sliki 9.10.



Slika 9.10: Graf funkcije hiperbolični sinus, narisan s pomočjo novodefinirane funkcije *graf_funkcije*

9.2.6 Primeri uporabe Matlaba

V tem razdelku bomo prikazali uporabnost Matlaba kot orodja za reševanje realnih problemov na nekaj primerih.

Prikaz prileganja kvadratne funkcije podatkom

Na neki datoteki z imenom A imejmo podatke o točkah v ravnini. V vsaki vrstici imamo koordinati x in y ene točke. Zanima nas, katera kvadratna funkcija z enačbo $y = ax^2 + bx + c$ se najbolj prilega tem točkam. Radi pa bi tudi točke in iskano kvadratno funkcijo narisali ter izračunali povprečno kvadratično oddaljenost točk do te kvadratne funkcije.

Matlab ima zelo uporaben ukaz `load`, s katerim vsebino datoteke preberemo v matriko:

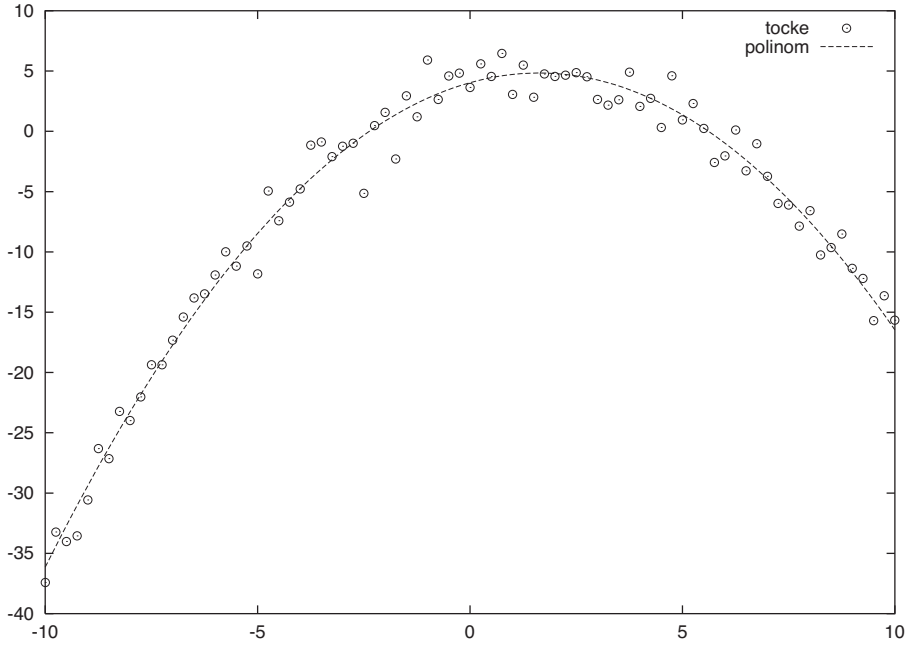
```
>> load A
```

S tem smo iz datoteke z imenom *A* prebrali podatke o točkah, ki so sedaj shranjene v matriki `A`. Nato pa lahko pokličemo funkcijo `polinom`, ki bo poskrbela za izračun parametrov a, b in c kvadratne funkcije in jo izrisala skupaj s točkami. Spodaj je primer kode funkcije `polinom`:

```
function dev = polinom(A,n,nft)
% function narisi(A,n,nft)
% za tocke v matriki A izracuna polinom stopnje n,
% ki se tem tockam najbolje prilega, ter narise tocke
% in polinom na isti graf. Polinom narise s pomocjo
% ravnih crt med nft tockami.

if nargin < 3 % ce tretji argument ni podan, zanj privzamemo vrednost 100
  nft = 100;
end

X = A(:,1);    % x koordinate se nahajajo v prvem stolpcu
Y = A(:,2);    % y koordinate pa v drugem

P = polyfit(X,Y,n);   % poiscemo polinom n-te stopnje, ki se
                      % najbolje prilega tockam
x1 = min(X);
x2 = max(X);
dx = (x2 - x1) / nft;
Xp = x1:dx:x2; % razdelimo interval med x1 in x2 na nft tock
Yp = polyval(P,Xp);

hold on;            % vsi grafi se risejo na isto sliko
plot(X,Y,"*;tocke;");  % narisemo tocke ...
plot(Xp,Yp,";polinom;");     % ... in polinom
hold off;

D = Y - polyval(P,X); % izracunamo se povpr. kvadr. napako
np = size(A);
dev = D'*D/ np(1);
```

Funkciji smo poleg matrike `A`, ki vsebuje podatke o točkah, dodali še parameter `n`, ki pove stopnjo polinoma, ki ga želimo izračunati, ter paramter `nft`, ki pove, koliko bo delilnih točk med črtami, s katerimi bomo približno narisali graf polinoma. Ta parameter je opcijski (kar pomeni, da ga lahko pri klicu funkcije podamo ali pa tudi ne). Funkcija lahko preko globalne spremenljivke `nargin` ugotovi, koliko parametrov ji je bilo podano. V primeru, da tretji argument ni bil podan, bo imela `nargin` vrednost 2, to pa v funkciji izkoristimo zato, da parameter `nft` postavimo na neko privzeto vrednost. Tretji argument pri klicu funkcije

`plot` podaja način, na kakršnega bodo točke narisane. Znak ‘‘`*`’’, pomeni, da bodo točke narisane posamezno, brez medsebojnega povezovanja z ravnimi črtami, z besedo med podpičjema pa Matlabu povemo, katero besedo naj uporabi v legendi grafa namesto privzete besede ‘‘`line`’’. Klic funkcije `polinom(A,2)`, pri čemer matrika `A` vsebuje nekaj točk, ki se približno prilegajo neki kvadratni funkciji, nam izriše graf na sliki 9.11. Omenimo še ukaz `hold`, s katerim lahko Matlab prisilimo v to, da bo vse grafe izrisal skupaj. To dosežemo z ukazom `hold on`. Z ukazom `hold off` pa Matlab vsak ukaz za risanje izvrši na novo, pri čemer se stvari, izrisane s predhodnimi ukazi, izbrišejo (tako je tudi privzeto obnašanje Matlaba).



Slika 9.11: Izris točk in grafa kvadratne funkcije

Risanje superkvadrikov

Superkvadriki [2] so geometrijska telesa, pri katerih je vsaka točka na površini definirana takole:

$$\begin{aligned} x &= a_1 \cos^{\epsilon_1} \eta \cos^{\epsilon_2} \omega, \\ y &= a_2 \cos^{\epsilon_1} \eta \sin^{\epsilon_2} \omega, \\ z &= a_3 \sin^{\epsilon_1} \eta. \end{aligned} \tag{9.2}$$

Spremenljivka η lahko zavzame vrednosti med $-\pi/2$ do $\pi/2$, ω pa med $-\pi$ in π. Tako so definirane točke na površini superkvadrika. Parametri

a_1, a_2 in a_3 določajo velikost superkvadrika, ϵ_1 in ϵ_2 obliko, imamo pa še parametre, ki določajo premik središča superkvadrika in kot rotacije okrog posameznih osi. Naloge se v Matlabu lotimo tako, da izračunamo koordinate nekaterih točk in jih narišemo v mrežo s pomočjo funkcije `mesh`. Spodaj je primer kode funkcije `sq`, ki nariše poljuben superkvadrik:

```
function sq(a1,a2,a3,e1,e2,C,rotX,rotY,rotZ)
% function sq(a1,a2,a3,e1,e2, [C, rotX, rotY, rotZ])
% narise superkvadrik s parametri a1,a2,a3 (velikost superkvadrika),
% e1,e2 (oblika), C (sredisce superkvadrika) in rotX, rotY, rotZ (kot
% rotacije okrog posamezne osi).
% Poskrbimo za privzete vrednosti nekaterih parametrov.

if nargin < 9
  rotZ = 0;
  if nargin < 8
    rotY = 0;
    if nargin < 7
      rotX = 0;
      if nargin < 6
        C = [ 0 0 0 ];
      end
    end
  end
end

% izracunamo rotacijske matrike
Rx = [ 1 0 0;
           0 cos(rotX) -sin(rotX);
           0 sin(rotX) cos(rotX) ];

Ry = [ cos(rotY) 0 sin(rotY);
           0 1 0;
           -sin(rotY) 0 cos(rotY)];

Rz = [ cos(rotZ) -sin(rotZ) 0;
           sin(rotZ) cos(rotZ) 0;
           0 0 1];

Rot = Rx*Ry*Rz;

% konstanta N doloca gostoto crt pri risanju mreze superkvadrika
N = 20;
```

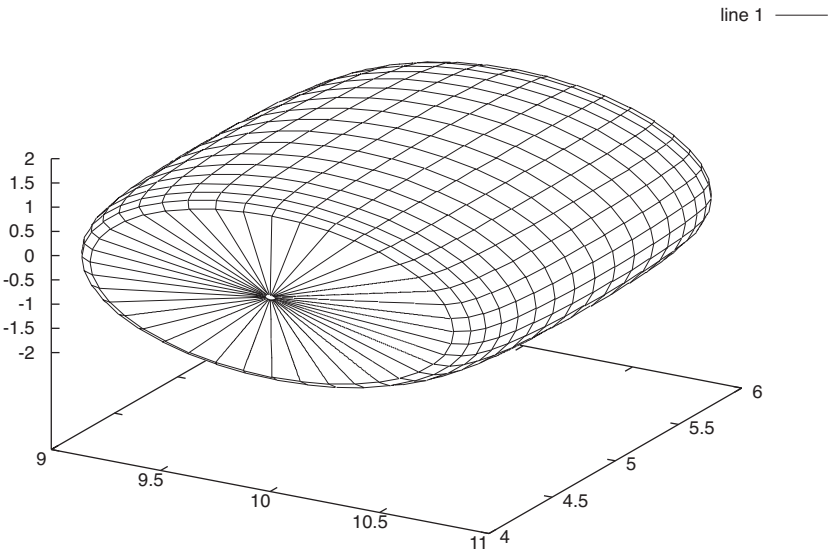
```
% izracunamo mrezo tock na povrsini superkvadrika
I = 0:N;
K1 = cos(-pi/2+I*pi/N);
K2 = sin(-pi/2+I*pi/N);
I = 0:2*N;
K3 = cos(-pi+I*pi/N);
K4 = sin(-pi+I*pi/N);

Xn=a1*(sign(K1).*exp(e1*log(abs(K1))))'*(sign(K3).*exp(e2*log(abs(K3))));
Yn=a2*(sign(K1).*exp(e1*log(abs(K1))))'*(sign(K4).*exp(e2*log(abs(K4))));
Zn=a3*(sign(K2).*exp(e2*log(abs(K2))))'*(1+zeros(1,2*N+1));

% mrezo tock se rotiramo in premaknemo za vektor C
X = C(1) + Rot(1,1)*Xn+Rot(1,2)*Yn+Rot(1,3)*Zn;
Y = C(2) + Rot(2,1)*Xn+Rot(2,2)*Yn+Rot(2,3)*Zn;
Z = C(3) + Rot(3,1)*Xn+Rot(3,2)*Yn+Rot(3,3)*Zn;

% narisemo mrezo tock
mesh(X,Y,Z);
```

Pri tem smo uporabili veliko znanih operatorjev iz razdelka 9.2.2 in funkcijo `zeros(n,m)`, ki vrne ničelno matriko velikosti $n \times m$. S klicem funkcije `sq(1,2,1,0.1,1,[10 5 0],pi/2)` lahko sedaj narišemo superkvadrik, prikazan na sliki 9.12.

Računanje določenega integrala poljubne funkcije

Določeni integral poljubne zvezne funkcije se lahko približno izračuna tako, da se interval razdeli na delilne točke in sešteje ploščine trapezov, ki jih oklepajo točke $(x_i, 0)$, $(x_{i+1}, 0)$, $(x_i, f(x_i))$ in $(x_{i+1}, f(x_{i+1}))$, pri čemer so x_i delilne točke. Rezultat je tem natančnejši, čimbolj skupaj so delilne točke. Spodaj je primer kode funkcije `integral`:

```
function sum = integral(f,x1,x2,dx)
% function sum = integral(f,x1,x2,dx)
% izracuna doloceni integral funkcije f, ki je podana
% z nizom znakov f, od x1 do x2 z razmakom dx.
% Za neodvisno spremenljivko uporabimo niz znakov xx.
% Primer klica: integral("xx*sin(xx)",0,pi,0.05)

X = x1:dx:x2;         % vektor delilnih tock
S = size(X);
n = S(2);                 % stevilo delilnih tock
X(n+1) = x2; % dodamo se x2, ker ga lahko ni se v vektorju X
```



Slika 9.12: Superkvadrik narisan z Matlabom

```
fs1 = strcat("f1=",f,";"); % niz znakov, ki predstavlja ukaz s
fs2 = strcat("f2=",f,";"); % katerim se izracuna vrednost funkcije

sum = 0;
for i=1:n
  s1 = sprintf("%f",X(i));
  s2 = sprintf("%f",X(i+1));
  cm1 = strrep(fs1,"xx",s1);
  cm2 = strrep(fs2,"xx",s2);
  eval(cm1);
  eval(cm2);
  % dodamo ploscino trapeza med delilnima tockama i in i+1
  sum = sum + dx*(f1+f2)/2;
end
```

Poleg že znanih funkcij in operatorjev smo uporabili še nekatere, ki operirajo z nizi znakov. Funkcija `strcat` dobi kot parameter poljubno število nizov znakov. Kot rezultat vrne niz znakov, ki je sestavljen iz staknjenih nizov znakov, ki so ji bili podani kot parameter. S funkcijo `sprintf` lahko cela in realna števila pretvorimo v niz znakov, podobno kot v programskem jeziku C. Funkcija `strrep(s,x,y)` pa vrne kot rezultat niz znakov `s`, kjer so vse pojavitve podniza `x` zamenjane s podnizom `y`. Ideja funkcije je v tem, da zgradi nize znakov ‘‘`f1=sin(xx);`’’ in ‘‘`f2=sin(xx);`’’ (če je f enak nizu ‘‘`sin(xx)`’’) s

pomočjo katerih lahko v funkciji `eval` izračunamo vrednosti funkcije v delilnih točkah in s tem tudi ploščino posameznega trapeza. Če želimo izvedeti, koliko je določeni integral funkcije naravnega logaritma od 1 do 10, lahko to izvemo z ukazom `integral("log(xx)",1,10,0.1)`. Funkcija ni napisana optimalno in jo je mogoče precej izboljšati v smislu časa, potrebnega za izvajanje.

9.2.7 Naloge

1. Kaj izpiše Matlab po spodnjem ukazu?

   ```
   A = [ [ [4;2;7;1] [2 3 2 1;7 3 2 1;5 4 3 0]'][4 9 8 1]' ]
   ```

2. Kaj izpiše Matlab po ukazu [2 3; 0 1] .* [0 1; 1 0] ?

3. Kako bi v Matlabu sestavili spodnji vektor s čimmanj tipkanja in računanja na roke?

   ```
   [ 1 2 4 8 16 32 64 128 ... ]
   ```

   Zadnja vrednost v tem vektorju naj bo 2^{100}.

4. Kako bi v Matlabu s čimmanj tipkanja izračunali koeficiente polinoma, ki ima ničle v lastnih vrednostih matrike A?

5. Napišite datoteko sum.m, ki vsebuje funkcijo `sum(n)`, ki vrne naslednjo vsoto:
   $$\sum_{i=1}^{n} \sum_{j=i+1}^{n} \frac{i^2}{\sqrt{j}}.$$

6. Napišite datoteko nicla.m, ki vsebuje funkcijo `nicla(P, x1, x2)`, ki z metodo bisekcije izračuna vrednost ničle polinoma, ki se nahaja med točkama `x1` in `x2`. Za ničlo naj funkcija proglasi sredino iskalnega intervala, ko se le ta zoži na manj kot `0.0001`.

7. Napišite datoteko positive.m, ki vsebuje funkcijo `positive(A)`, ki vrne `1`, če so vse lastne vrednosti matrike A pozitivne in 0 sicer.

8. Napišite datoteko prastevilo.m, ki vsebuje funkcijo `prastevilo(n)`, ki vrne `1`, če je `n` praštevilo in 0 sicer.

9. Napišite datoteko mindiv.m, ki vsebuje funkcijo `mindiv(a, b)`, ki vrne največji skupni deljitelj števil `a` in `b`.

10. Napišite datoteko sort.m, ki vsebuje funkcijo `sort(V)`, ki kot rezultat vrne vektor, v katerem so števila iz vektorja V urejena po velikosti.

9.3 Koristne spletne povezave

1. Wolfram Research – spletna stran razvijalcev programa Mathematica:
   `http://www.wolfram.com/`
2. Strani namenjene Matlabu na University of Texas:
   `http://www-math.cc.utexas.edu/math/Matlab/`
3. Domača stran podjetja MathWorks, ki izdeluje Matlab:
   `http://www.mathworks.com/`
4. Domača stran programa Octave - brezplačne različice Matlaba:
   `http://www.octave.org`

9.4 Literatura

[1] D. M. Etter and D. M. Etter. *Introduction to MATLAB for Engineers and Scientists*. Prentice Hall, 1995.

[2] R. Gass. *Mathematica for Scientists and Engineers: Using Mathematica to do Science*. Prentice-Hall, New Jersey, 1998.

[3] A. Gray. *Modern Differential Geometry of Curves and Surfaces with Mathematica*. CRC Press, Boca Raton, second edition, 1998.

[4] A. Jaklič, A. Leonardis, and F. Solina. *Segmentation and Recovery of Superquadrics*. Kluwer, Dordrecht, 2000.

10

Medmrežje in svetovni splet

Internet ali medmrežje je od prvih začetkov v 60-ih letih 20. stoletja, ko se je imenoval Arpanet in je povezoval le nekaj ameriških univerz in raziskovalnih inštitutov, zrasel v medij, ki dejansko povezuje ves svet. K temu uspehu je v veliki meri pomagal protokol TCP/IP, ki omogoča delovanje omrežja brez centralnega nadzora. Delovanje brez centralnega nadzora oziroma centralnega vozlišča je bila pomembna funkcionalna zahteva njegovega naročnika, ameriškega obrambnega ministrstva, saj je to bilo še v času oboroževalne tekme med takratnimi supersilama in grožnje z medcelinskimi raketami oboroženimi z jedrskimi konicami. Internet je že od samega začetka omogočal izmenjavo elektronske pošte in delo na oddaljenih računalnikih (protokol telnet). S pojavom svetovnega spleta (WWW – World Wide Web) pa je internet stopil iz znanstvenih in strokovnih krogov v širšo javnost in se bliskovito razširil, saj s preprostim grafičnim vmesnikom, ki ga imajo današnji spletni brskalniki, lahko vsakdo sam išče po internetu raznovrstne informacije. Jedro spletnih strani predstavljajo dokumenti, napisani v jeziku HTML (*HyperText Markup Language*). To je jezik, ki omogoča povezovanje med točkami (na primer besedami) v besedilu z drugimi besedili, slikami, zvokom ali video sekvencami, ki so lahko na istem računalniku ali pa kjerkoli na svetovnem spletu. Ta princip povezljivosti se imenuje **hipertekst**.

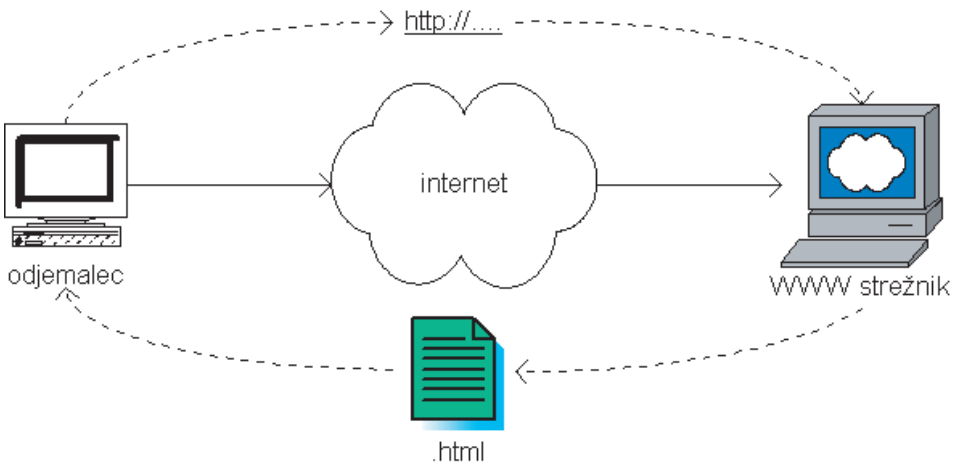
Pojem hiperteksta je definiral Ted Nelson leta 1965. O samem konceptu hipertekstovnih povezav pa je v svojem projektu Memex govoril že Vannevar Bush leta 1945. Prva uspešna izvedba hipertekstovnega sistema je bil Applov Hypercard leta 1987. Jezik HTML, ki ga je leta 1991 razvil Tim Berners-Lee v jedrskem raziskovalnem centru CERN v Ženevi, pa je možnost povezav med točkami v besedilih, slikah, zvočnih in video zapisih, in to neodvisno od tega, kje se ti zapisi nahajajo, ponesel na internet. Princip hipertekstovnih povezav je tesno povezan tudi z uporabo večpredstavitvenih vsebin (angl. *multimedia*), saj je možno med seboj povezati različne medije, ne le tekstovne informacije.

Poleg interaktivnosti in grafičnega uporabniškega vmesnika pa je k uspehu svetovnega spleta pripomogla tudi neodvisnost od strojne in sistemske računalniške opreme. Za uporabo svetovnega spleta ni pomembna ne vrsta računalnika, tip procesorja ali operacijskega sistema. Dovolj je le to, da ima računalnik dostop do interneta in da je na računalniku nameščen spletni brskalnik.

V nadaljevanju poglavja je najprej nekaj osnovnih navodil o izdelavi spletnih strani. Nato sledi obravnava spletnih brskalnikov in kako se išče informacije na spletu ter uporaba elektronske pošte. Nazadnje pa obravnavamo jezika HTML in XML, ki sta osnova vseh spletnih strani.

10.1 Kako deluje svetovni splet

Svetovni splet je zasnovan na internetnih protokolih. Iz uporabniškega vidika si lahko predstavljamo delovanje spleta kot interakcijo med odjemalci in strežniki, ki poteka prek internetnih povezav (glej sliko 10.1).



Slika 10.1: Odpiranje spletne strani

Odjemalec je tipično spletni brskalnik, ki teče na računalniku uporabnika. Brskalnik iz naslova v obliki URL pridobi naslov strežnika in iz njega internetni naslov (angl. *IP address*). Potem vzpostavi internetno povezavo do strežnika in pošlje zahtevek, ki vsebuje URL naslov zahtevanega dokumenta in odvisno od uporabljenega protokola še nekaj dodatnih informacij. V primeru protokola HTTP brskalniki pošljejo še številne informacije o brskalniku in uporabnikovem računalniku, kot so način povezave, različica brskalnika in operacijskega sistema, zaželjeni jezik dokumentov, tipi sprejemljivih dokumentov, sprejemljivi nabori znakov in naslov povezave, ki nas je pripeljala do zahtevanega dokumenta.

Strežnik prebere zahtevek odjemalca in odgovori z glavo in vsebino dokumenta (če seveda najde ustrezen dokument). V primeru protokola HTTP odgovori s statusno kodo in z različico protokola, ki ga uporablja. Običajno vrne tudi datum in uro dokumenta, različico strežnika in dodatkov, URL vrnjenega dokumenta, način povezave, način kodiranja prenosa in tip ter znakovni nabor vsebine. V kolikor strežnik najde ustrezen dokument, sledi vsebina, v nasprotnem primeru javi odjemalcu kodo napake in, odvisno od konfiguracije, ustrezno obvestilo, nadomestno stran ali drug naslov.

Odjemalec (spletni brskalnik) nato interpretira vsebino dokumenta in jo prikaže na zaslonu.

Poleg protokola HTTP na spletu pogosto srečamo tudi varnejšo različico HTTPS, ter protokol za prenos datotek FTP, v redkih primerih pa tudi še starejši protokol Gopher. Svetovni splet sestavljajo vsi dokumenti, ki so dosegljivi preko internetnih protokolov in so medsebojno povezani preko imenovanih povezav (URL) v dokumentih.

10.2 Izdelava spletnih strani

Izdelava spletnih strani zahteva poleg tehničnega znanja, kot je programiranje z jezikom HTML ali z drugimi vizualnimi orodji za oblikovanje spletnih strani tudi skrb za vsebino spletnih strani, kar pomeni pravilen izbor besedil ter grafične in zvočne opreme. Vsebina spletne strani morajo biti ustrezno strukturirana za medij medmrežja, grafika in zvok pa morata biti optimizirana za hiter prenos po računalniškem omrežju. Vse skupaj, besedila in slikovna oprema, mora biti ustrezno oblikovano in povezano v tako strukturo, po kateri se je možno hitro premikati in poiskati določene informacije. Izdelava spletnih strani zahteva dobro koordinacijo dela, saj so za večje spletne predstavitve potrebne skupine različnih strokovnjakov (piscev besedil, grafičnih oblikovalcev, programerjev). Spletne strani je potrebno tudi redno vzdrževati, kar pomeni sproti obnavljati informacije na spletnih straneh, in jih po potrebi tudi tržiti.

Sam postopek izdelave spletne predstavitve je možno združiti v naslednje korake:

1. Določitev ciljnih uporabnikov spletne strani, to so lahko:
   - spletni deskarji in naključni uporabniki ali ciljna publika (naše stranke, kupci itd.),
   - začetniki ali izkušeni uporabniki.
2. Kaj je osnovni namen spletne strani (na primer informiranje, prodaja, zabava ipd)?

3. Na osnovi namena oblikovanje konkretnih ciljev (na primer podpora strankam, informiranje zaposlenih ipd).
4. Določitev obsega informacij na spletni strani (koliko besedila, koliko slik).
5. Kakšen bo osnovni način komuniciranja z obiskovalci spletne strani (tekstovni, vizualni)?
6. Na osnovi obsega vsega gradiva in načina komuniciranja se določi strukturo spletne strani.
7. Na osnovi strukture spletne strani se določi primeren način navigacije po spletni strani.
8. Sledi grafično oblikovanje spletne strani.
9. Izvedba spletne strani.
10. Vzdrževanje spletne strani.

10.2.1 Pisanje za svetovni splet

Pri pisanju besedil za branje na računalniškem zaslonu moramo upoštevati osnovno zakonitost, da je branje z zaslona počasnejše kot branje s papirja. Kljub novim tehnologijam računalniških zaslonov in njihovi višji ločljivosti je branje s papirja še vedno lažje predvsem zaradi višjega kontrasta tiska na papirju pri različnih svetlobnih pogojih in zaradi večje fleksibilnosti papirnatega medija. Danes žal še ne obstaja zaslon, ki bi ga lahko naprimer zložili in vtaknili v žep kopalk, in ga po kopanju razprli in pod močno sončno svetlobo na njem prebrali besedilo.

Pri branju informacij na računalniku bralci računalniški zaslon najprej preletijo podobno kot pri branju časopisa, šele nato začno z branjem. Težišče informacije, podane preko zaslona, mora biti zato v vizualni organizaciji strani in v slikovnih informacijah. Komuniciranje preko računalniškega zaslona je zelo neposredno. Tipični uporabnik posveti svojo pozornost eni spletni strani v povprečju le okoli osem sekund. Zato je potrebno najvažnejše informacije podati hitro in na najbolj vidnem mestu (na vrhu strani, z veliko in krepko pisavo)! Tudi v globino spletne strani se tipični uporabnik sprehodi le do dva pritiska na miško daleč. Informacije morajo biti zato strukturirane v majhne pakete, da posamezne strani niso predolge in da ni potrebno predolgo listanje.

Za svetovni splet moramo pisati natančno in jasno, tako da uporabljamo kratke stavke ter jasne in aktivne glagole. Besedilo poskušamo jasno strukturirati s pomočjo spiskov ali tabel. Ne uporabljajmo žargona, fraz in kratic, ki niso razumljive širši javnosti. Zavedati se moramo, da je za

svetovni splet javnost lahko kar cel svet. Pri pisanju in strukturiranju spletne predstavitve si je pametno vnaprej postaviti splošne omejitve, kot so na primer največje število pritiskov na miško do določene vrste informacij, povprečna dolžina besedila o določeni vrsti vsebine, dolžina odstavkov (na primer 5 vrstic). Pri uporabi besedil drugih avtorjev moramo paziti na avtorske pravice, saj je za namen dobesednega citiranja dovoljeno navesti le do 250 besed.

10.2.2 Slikovna oprema spletnih strani

Zaradi neposrednosti spletnega medija je uporaba slikovnih informacij [5, 6, 7] zelo zaželena. Numerične vrednosti lahko predstavimo kot:

- kolač kadar gre za dele celote,
- histogram če želimo predstaviti razmerja med vrednostmi.

Kvalitativne informacije lahko predstavimo v obliki

- urnika ali tabele,
- potek dogajanja kot diagram s puščicami,
- organizacijo ali hierarhijo z drevesnim diagramom.

S pomočjo tabel in spiskov je možno razčleniti in nadgraditi besedilo. S pomočjo puščic, vodorovnih in navpičnih črt in različnih barv ozadja pa je možno hitreje voditi pogled bralca med posameznimi informacijskimi enotami. S slikami lažje predstavimo razne koncepte. Možnosti segajo od tehničnih risb do ikon, ki jih veliko uporabljamo v grafičnih uporabniških vmesnikih.

Pri uporabi slikovnega gradiva moramo upoštevati načelo, da naj grafična oprema ne le polepša, temveč predvsem izboljša informacijo. Izbrati moramo takšno grafično obliko, ki na najbolj preprost in jasen način ilustrira določeno informacijo. Pomembna in pogosto prezrta možnost grafičnega oblikovanja je prazen prostor. S pomočjo spuščanja vrstic, odmikanja in zamikanja objektov na zaslonu lahko dosežemo velik učinek.

Nenazadnje moramo upoštevati pri uporabi slik na svetovnem spletu tudi nekatere tehnične vidike, kot so čas nalaganja in barvna ločljivost. Najbolj razširjeni grafični formati, ki se uporabljajo na spletu, so:

- GIF (256 barv), ki je primeren za risbe in diagrame,
- JPEG, ki omogoča 24-bitne barve in različne stopnje stiskanja ter je zato najbolj primeren za fotografije, ter

- PNG, ki omogoča brezizgubno stiskanje slik s poljubno barvno globino in v zadnjem času tudi zaradi težav s patenti nadomešča format GIF.

10.2.3 Komuniciranje z obiskovalci spletnih strani

Na spletni strani naj bi bile informacije podane v obliki dialoga, ki obiskovalca strani zvabi v interakcijo. Komunikacija naj bi bila intimna, s tem da se prilagodi uporabnikovim potrebam in pričakovanjem. Na obiskovalca se obračamo v drugi osebi in tako kot v pogovoru uporabljamo le kratke stavke ali celo le dele stavkov. Pri naslavljanju obiskovalcev ne smemo delati omejujočih predpostavk, saj ne vemo vnaprej, kdo so, koliko so stari, kakšno izobrazbo imajo, ali dobro razumejo jezik spletne strani.

Z uporabnikovega gledišča obsega interaktivno komuniciranje navodila za ravnanje, prilagoditev prezentacije ter odgovor na zahtevo. Pri interaktivnem komuniciranju moramo biti vljudni ter upoštevati uporabnikove odgovore. Interakcija mora biti za uporabnika tehnično nezahtevna. Odgovore obiskovalcev lahko klasificiramo tako, da že vnaprej predvidimo najbolj značilne odgovore in tako grupiramo uporabnike. Pomagamo jim lahko z vnaprej pripravljenimi odgovori na najbolj pogosta vprašanja (angl. *FAQ – Frequently Asked Questions*). Narava računalniške tehnologije pa omogoča tudi povsem individualizirano obravnavo uporabnikov. Spletna trgovina si lahko zapomni, kaj smo že prej kupili, katero številko obleke nosimo, katero zvrst glasbe poslušamo, katere avtorje beremo, ter na podlagi teh informacij oceni, kaj nas bo najverjetneje zanimalo v prihodnosti. Za interakcijo uporabljamo označevanje hipertekstovnih točk, izbiranje ukazov iz menijev, vnos podatkov v vnaprej pripravljena polja, ikone in povezave.

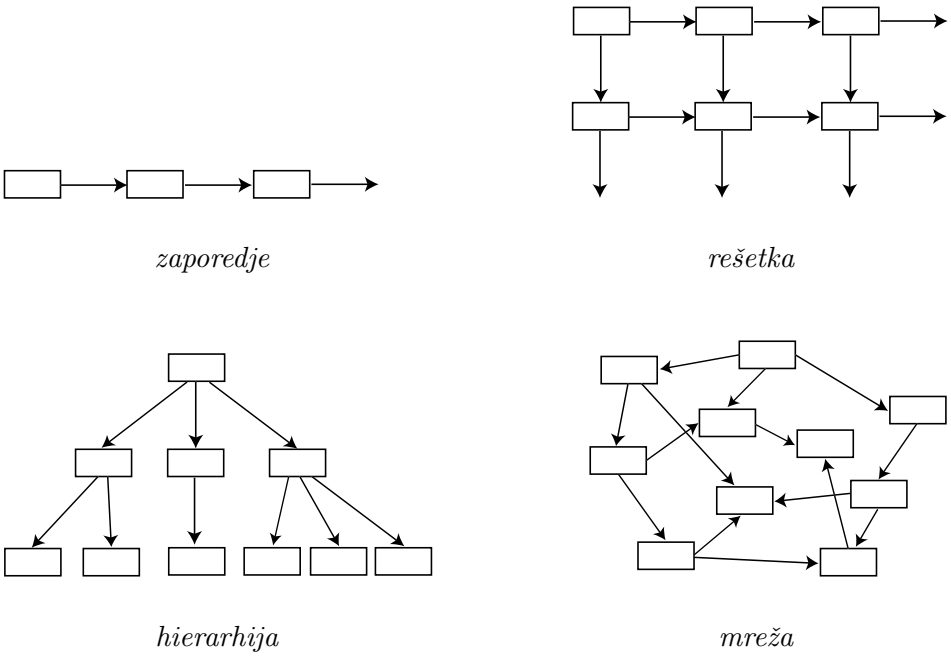
Poseben vidik interakcije je zbiranje informacij o obiskovalcih strani, ki je lahko do določene mere povsem avtomatsko in ki nam omogoča, da lahko našo spletno predtavitev še bolj prilagodimo tipičnim obiskovalcem.

10.2.4 Struktura spletnih strani

Spletne strani lahko strukturiramo na različne načine. Ločimo štiri osnovne strukture spletnih strani (slika 10.2): zaporedje, rešetko (angl. grid), hierarhijo (na primer drevesno) in mrežo (angl. web). Od strukture spletne strani je odvisno, kako hitro lahko najdemo določeno informacijo.

Strukturo spletne predstavitve moramo izbrati glede na logično strukturo informacij (tabela 10.1):

- za **urjenje in izpite** moramo omejiti gibanje na naprej - nazaj,



Slika 10.2: Različne strukture spletnih strani

struktura	***čas interakcije***	
	kratek	*dolg*
linearna	urjenje	poučevanje
nelinearna	viri	izobraževanje

Tabela 10.1: Izbira strukture spletne strani glede na njen namen

- za **poučevanje** so poleg povezave naprej pomembne tudi povezave v globino na podrobnejše razlage,
- za **izobraževanje** je potrebna večja fleksibilnost, ki jo pričakujejo zahtevni uporabniki,
- pri **virih** je pomembno predvsem hitro iskanje določenih informacij (na primer telefonskih številk v telefonskem imeniku ali razlago besed v slovarju).

10.2.5 Navigacija po spletnih straneh

Osnovno pravilo za navigacijo po spletnih straneh je, da mora omogočiti dostop do vseh delov spletne predstavitve. Spletne strani, na katere ne kaže nobena povezava, imenujemo mrtve. Obiskovalec naj bi ves čas vedel, kje na spletni predstavitvi se nahaja. Zato naj bi imela vsaka spletna stran v svoji glavi ali nogi osnovne povezave naprej in nazaj po

predstavitvi ter nazaj na svojo začetno stran. Pri preiskovanju spletnih strani moramo ločiti med absolutnimi (glede na spletno predstavitev) in relativnimi povezavami (vrstni red preiskovanja v brskalniku). Hitrost dostopa do neke informacije merimo s številom klikov, ki so potrebni, da pridemo do nje z začetne strani. Obsežnejše spletne predstavitve morajo imeti kazalo in možnost iskanja. Koristni so tudi odgovori na pogosta vprašanja (angl. *FAQ*) in kazalci na novosti.

10.2.6 Oblikovanje spletnih strani

Pri oblikovanju spletnih strani moramo izbrati pravo mero in poiskati tak vizualni izgled, ki bo podprl vsebinsko strukturo informacij. Osnovni vizualni izgled dosežemo z ravnotežjem praznih in polnih delov strani. Spletne strani, ki sestavljajo neko spletno predstavitev, morajo biti enotno oblikovane. Zato je smiselno oblikovati vzorec (angl. *template*) za tipične vrste spletnih strani. Na vzorcu določimo razporeditev besedila in slik, določimo vrsto in velikost pisav za naslove in podnaslove, oblikujemo ikone, ki jih bomo uporabljali v celotni spletni predstavitvi. Standardne povezave (naprej, nazaj, na začetno stran) morajo biti vedno na istem mestu. Najvažnejše informacije morajo biti na vrhu strani. Strani, kjer je veliko povezav, naj bi bile kratke, tako da pomikanje po strani ni potrebno. Vsaka spletna predstavitev naj bi imela navedenega avtorja oziroma institucijo, datum zadnje spremembe, kontakt (e-pošta, telefon), definirane avtorske pravice (ali se vsebina sme kopirati ali je zaščitena), URL in ime dokumenta.

Z oblikovalskega vidika je največji problem pri oblikovanju spletnih strani to, da je izgled strani odvisen od vrste brskalnika in zaslona oziroma terminala (ločljivost, barve, hitrost dostopa). Jezik HTML se razvija in dopolnjuje, da bi bilo oblikovanje spletnih strani lažje in bolj fleksibilno, vendar pa žal kljub naporom po standardizaciji vsi spletni brskalniki ne podpirajo vseh razširitev jezika HTML v enaki meri [8]. Standard XML podobno kot LaTeX omogoča ločitev vsebine in oblike, kar omogoča lažje in glede na vsebinsko strukturo povsem uniformno oblikovanje spletnih strani.

Branje dolgih besedil z računalniškega zaslona je utrudljivo. Zato daljša besedila namesto v formatu HTML raje shranimo v PostScript ali v formatu PDF (Portable Document Format), saj jih uporabniki običajno raje natisnejo in berejo s papirja. V formatu PDF shranjeno in nato natisnjeno besedilo bo točno v taki obliki, s tako pisavo in črkami, kot jih je zapisal in oblikoval avtor. Zato v formatu PDF pogosto shranjujemo tudi razne obrazce in druga besedila, kjer se mora oblika natančno ohraniti.

Za oblikovanje spletnih strani obstaja danes že veliko različnih orodij,

predvsem vizualnih urejevalnikov, tako da ni več potrebno kodiranje neposredno z jezikom HTML. Če uporabljamo vizualni urejevalnik, pa ne smemo pozabiti, da uporabniki ne bodo nujno videli istega, kar vidimo mi! Vizualni urejevalniki spletno stran tudi shranijo v datoteko HTML, ki jo interpretira spletni brskalnik. Videz spletne strani je vedno odvisen od vrste brskalnika in terminala. Prav zaradi pestrosti današnjih spletnih terminalov (mobilni telefon, dlančnik ipd) si želimo ločiti vsebino in obliko, kar omogoča standard XML. Tako se lahko videz spletne strani prilagodi vrsti terminala ali hitrosti dostopa.

Nekatere najpogostejše napake pri oblikovanju spletnih strani so:

- nerodne povezave, kot na primer: klikni sem,
- povezave na neobstoječe strani,
- napačne ali zastarele informacije,
- dokumenti, ki so oblikovani za določeno širino in višino zaslona,
- celotno besedilo napisano v krepki pisavi ali z velikimi črkami,
- moteče ozadje za besedila,
- neusklajene barve.

10.2.7 Vzdrževanje spletnih strani

Spletne strani moramo redno vzdrževati. Pri vzdrževanju spletnih strani je najpomembneje, da definiramo in načrtujemo proces vzdrževanja, to je kdo, kdaj in kako bo delal spremembe. Pri vzdrževanju moramo ločiti med različnimi vrstami revizij:

- periodični popravki (dnevni, tedenski, mesečni, ...),
- dodajanje novih funkcij in elementov (po potrebi in občasno),
- popolnoma novo oblikovanje (vsako leto ali dve).

Kot že pri osnovnem oblikovanju spletnih strani moramo izhajati iz vzorcev za posamezne vrste dokumentov. Pri administraciji moramo skrbno izbrati način poimenovanja datotek iz katerih je sestavljena spletna stran. Pomemben vidik vzdrževanja je tudi preverjanje, če vse povezave, tudi na tuje spletne strani, delujejo.

Internet in svetovni splet se nezadržno širita. Vse države imajo danes svoje nacionalne domene, poleg tega pa obstaja še nekaj globalnih domen (npr. `.com, .edu, .gov, .org`), v katerih pa že zmanjkuje komercialno zanimivih imen. Zato načrtujejo nove globalne domene, problem pa

postaja tudi omejen naslovni prostor za številke IP. Vse oblike komunikacij (podatkovne, govorne, mobilne) se vse bolj integrirajo. Jezik HTML se dopolnjuje, hkrati pa je vedno težje obdržati enoten standard med različnimi brskalniki. Pojavljajo se novi zapisi slik, videa in zvoka. Vedno težje je obvladovati obseg informacij na svetovnem spletu, saj je veliko spletnih strani povezano z obsežnimi podatkovnimi bazami, tako da se pri vsaki spremembi v podatkovni bazi ali pri poizvedbi v podatkovni bazi avtomatsko generira koda HTML. Potrebujemo boljše iskalnike, ki bodo znali rangirati in klasificirati informacije na internetu.

10.3 Brskalnik in odjemalec za e-pošto

Spletni brskalnik (angl. *web browser*) je samostojna računalniška aplikacija, ki poskrbi za dve stvari:

- Komunikacijsko povezavo preko interneta s spletnim strežnikom, od katerega zahteva spletno stran, in prenos spletne strani na lokalni računalnik. Povezavo spletnega brskalnika s spletnim strežnikom predpisuje protokol HTTP.
- Tolmačenje oznakHTML, ki opisujejo spletno stran, in prikaz spletne strani na zaslonu v skladu z oblikovnimi lastnostmi, ki jih označbe HTML predpisujejo.

Danes obstaja cela vrsta spletnih brskalnikov, kot so Mozilla, Netscape, MS Internet Explorer, Apple Safari, Opera itd. V nadaljevanju si bomo ogledali delo z brskalnikom Mozilla.

10.3.1 Brskalnik Mozilla

Na sliki 10.3 se nahaja spletni brskalnik Mozilla, različica 1.7.2. Mozilla je spletni brskalnik in zbirka orodij z odprto izvorno kodo, ki ga odlikuje podpora standardom, zmogljivost in prenosljivost. Mozilla.org izdeluje programske pakete za preizkušanje in pridobivanje povratnih informacij. Nekateri opisi in nastavitve v nadaljevanju se bodo nanašali prav na to različico, vendar pa so osnovni koncepti dela z vsakim spletnim brskalnikom zelo podobni. V operacijskem sistemu Linux poženemo brskalnik Mozilla iz terminalskega okna z ukazom:

```
mozilla &
```

Uporabniški vmesnik sestavljajo glavni meni, ena ali več orodnih vrstic, okno z vsebino spletne strani in statusna vrstica.

Če želimo prikazati vsebino določene spletne strani, moramo poznati njeno oznako URL. URL (angl. *Uniform Resource Locator*) je standard za

Slika 10.3: Spletni brskalnik Mozilla

naslavljanje dokumentov. Gre za enolično oznako spletne strani, ki je sestavljena na sledeči način:
`protokol://uporabnik:geslo@naslov_strežnika:vrata/mapa/datoteka`

`protokol` je oznaka internetnega protokola. Če dostopamo do spletne strani na spletnem strežniku, je ta `http` za običajni ali pa `https` za varni HTTP (angl. *secure HTTP*). `naslov_strežnika` je naziv ali številka IP mrežnega vmesnika računalnika, na katerem je nameščen spletni strežnik. Številka vrat, na katerih praviloma posluša spletni strežnik, je `80` in jo lahko izpustimo. Prav tako izpustimo ime uporabnika in geslo, če strežnik tega ne zahteva (običajno zahteva prijavo strežnik FTP). Na sliki 10.3 dostopamo do strani na računalniku `mozilla.lugos.si`, zato je njena oznaka URL `http://mozilla.lugos.si/index.php` .

Svoj URL imajo tudi datoteke na lokalnem računalniku. Ta način dostopa bomo uporabljali v primerih spletnih strani v razdelku 10.4. URL datoteke `test.html` na imeniku `/home/users/upo/JanezNovak` je: `file:///home/users/upo/JanezNovak/test.html` .

Vsebino spletne strani prikažemo tako, da vpišemo njen URL v vnosno polje, ki se nahaja v orodni vrstici. Podobno lahko storimo z menijem "Datoteka→Odpri spletno mesto ...", ki prikaže pogovorno okno za vpis oznake URL. Če v tem pogovornem oknu izberemo "Izberi datoteko ...", lahko v novem pogovornem oknu poiščemo datoteko na lokalnem računalniku. Novo spletno stran prikažemo tudi tako, da na trenutno prikazani strani kliknemo na povezavo, ki kaže nanjo. V vseh omenjenih primerih se bo brskalnik povezal s spletnim strežnikom in prenesel vsebino spletne strani na lokalni računalnik. V primeru, da se je od trenutka zadnjega prenosa vsebina spletne strani že spremenila, lahko z gumbom "Ponovno naloži" iz orodne vrstice stran ponovno preberemo. Če prenos strani traja predolgo, ga lahko prekinemo z gumbom "Ustavi".

Če se brskalnik ne more povezati s strežnikom, bo prikazal ustrezno sporočilo. Med stranmi, ki smo jih že pregledali, se pomikamo z gumboma iz orodne vrstice "Nazaj" in "Naprej". Vsebino spletne strani natisnemo z gubmom "Natisni". Kodo HTML spletne strani, ki jo trenutno preglejujemo, prikažemo z ukazom iz menija "Pogled→ Izvorna koda strani". Spletne strani, ki jih pogosto obiskujemo, lahko dodamo med zaznamke. Zaznamke dodajamo in urejamo z izbirama "Dodaj stran med zaznamke" in "Upravljanje zaznamkov ..." gumba "Zaznamki" ali iz menija "Zaznamki".

10.3.2 Iskanje po svetovnem spletu

Za deskanje po svetovnem spletu moramo poznati oznake URL spletnih strani, ki jih želimo obiskati. Največkrat poznamo le tematiko ali ključne besede, ki bi jih nekje na spletu radi poiskali, ne vemo pa, kje se tovrstne vsebine nahajajo. V tem primeru si pomagamo z iskalniki, ki nam olajšajo deskanje po nepregledni in kaotični množici spletnih strani v svetovnem spletu.

Iskalnik ali spletni indeks je ogromna podatkovna zbirka, ki vsebuje podatke o straneh v svetovnem spletu. Zgrajen je s pomočjo posebnih programov (angl. *crawler, robot program*), ki se neprenehoma sprehajajo po svetovnem spletu in prinašajo spletne strani. Dnevno obiščejo tudi več kot 10 milijonov strani. Te se pregledajo in indeksirajo po različnih kriterijih. Eden najpomembnejših je navadno naslov spletne strani. Če želimo, da tovrstni program obišče tudi našo spletno stran, mora obstajati čim več povezav nanjo. Lahko pa jo v določeni indeks vključimo tudi ročno. Uporabniki vidimo iskalnik običajno kot spletno aplikacijo, ki nam

omogoča iskanje po podatkovni zbirki spletnih strani. Poženemo ga v brskalniku, tako da v vnosno polje orodne vrstice vpišemo njegov URL.

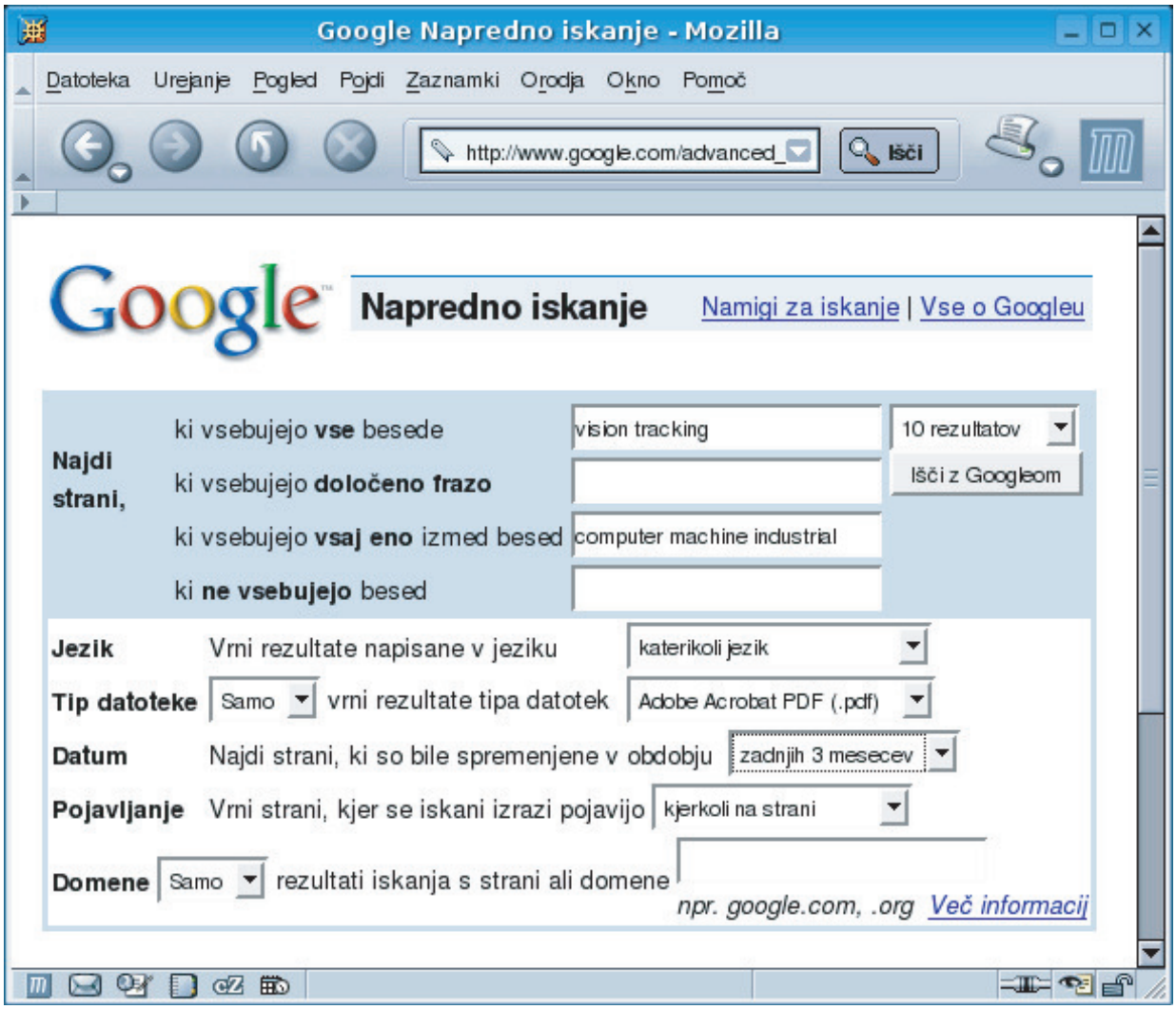
Najbolj priljubljeni spletni iskalniki so Google (`www.google.com`), Altavista (www.altavista.com), Yahoo (`www.yahoo.com`), Lycos (`www.lycos.com`), Najdi.si (`www.najdi.si`), Matkurja (`www.matkurja.com`) itd. Osnovni principi iskanja so podobni. Vsi iskalniki imajo nekje na spletni strani vnosno polje, v katerega vpišemo kriterije iskanja, gumb, s katerim iskanje sprožimo, kot rezultat pa nam vrnejo spisek spletnih strani, ki ustrezajo zahtevanemu kriteriju. V nadaljevanju bo opisano iskanje s trenutno najbolj razširjenim iskalnikom Google.

Za besedno iskanje po podatkovni bazi spletnih strani lahko s pomočjo iskalnika Google iščemo na dva načina. Za osnovni način iskanja vpišemo iskane besede v vnosno polje in pritisnemo gumb "Išči z Googleom". S pomočjo gumba "Poskusi srečo" pa nam iskalnik odpre najvišje rangirano povezavo. Napredno iskanje pa izberemo s pomočjo povezave "Napredno iskanje".

Osnovni način iskanja omogoča iskanje določenih ključnih besed ali besednih zvez. Besedne zveze se od besed ločijo tako, da jih vpišemo med narekovaje. Besede ali besedne zveze vpišemo v vnosno polje, iskanje pa poženemo z gumbom "Išči z Googleom". Iskalnik bo prikazal seznamom spletnih strani, ki vsebujejo vse iskane besede. Iskalnik namreč med navedenimi besedami sam doda operator AND, kar pomeni da bo iskal vse besede oziroma besedne zveze. Če hočemo poiskati strani, ki vsebujejo samo eno od navedenih besed moramo dodati operator OR. Zelo uporabna sta tudi znaka "+" in "-", ki ju postavimo pred besede. Iskalnik samodejno izključi iz iskanja najpogostejše besede kot so, vezniki, vprašalnice, izključi pa tudi enomestne številke in enojne črke. Če hočemo v iskanje vključiti katero izmed takšnih besed moramo pred to besedo dodati znak "+". Če pa hočemo, da najdene strani ne vsebujejo določene besede moramo pred to besedo dodati znak "-".

Pri naprednem iskanju, ki je prikazano na sliki 10.4, pa imamo na voljo štiri vnosna polja. V prvo polje vpišemo vse besede, ki bi jih radi našli, v drugo polje vpišemo iskano besedno zvezo oziroma frazo, v tretje napišemo besede med katerimi bi radi našli vsaj eno, v zadnje vnosno polje pa vpišemo besede, katere nočemo, da jih najdene strani vsebujejo. Iskanje lahko še dodatno omejimo glede na jezik v katerem je napisana spletna stran ali na vrsto datotete. Iščemo lahko samo strani, ki so bile spremenjene v določenem obdobju. Iskane besede lahko iščemo samo v določenem delu strani: v naslovu, v besedilu, v URL naslovu strani, v povezavi na stran. Iskanje lahko tudi omejimo samo na določeno domeno oziroma področje domen.

Na sliki 10.4 vidimo primer iskanja strani na temo sledenja z metodami



Slika 10.4: Zahtevnejši način iskanja v iskalniku Google

računalniškega vida, ki niso starejše od treh mesecev. V seznamu z rezultati iskanja želimo samo strani v formatu PDF.

Včasih pa hočemo na svetovnem spletu najti točno določeno stvar, kot je na primer slika, novica ali pa izdelek. Za takšno iskanje imamo na voljo specializirane iskalnike, ki jih lahko izberemo s pomočjo povezav nad vnosnim poljem iskalnika. Za iskanje slik izberemo “Slike”, za preiskovanje po forumih “Skupine”, če pa želimo iskati po imenikih, kjer so spletne strani organizirane po temah v kategorije izberemo “Imenik”.

10.3.3 Odjemalec za elektronsko pošto Mozilla Mail

Za pošiljanje in branje elektronske pošte potrebujemo uporabniško ime na strežniku za elektronsko pošto (angl. *mail server*) in odjemalec (angl. *mail client*), s katerim se na strežnik prijavimo.

Odjemalec za elektronsko pošto je računalniška aplikacija, ki:

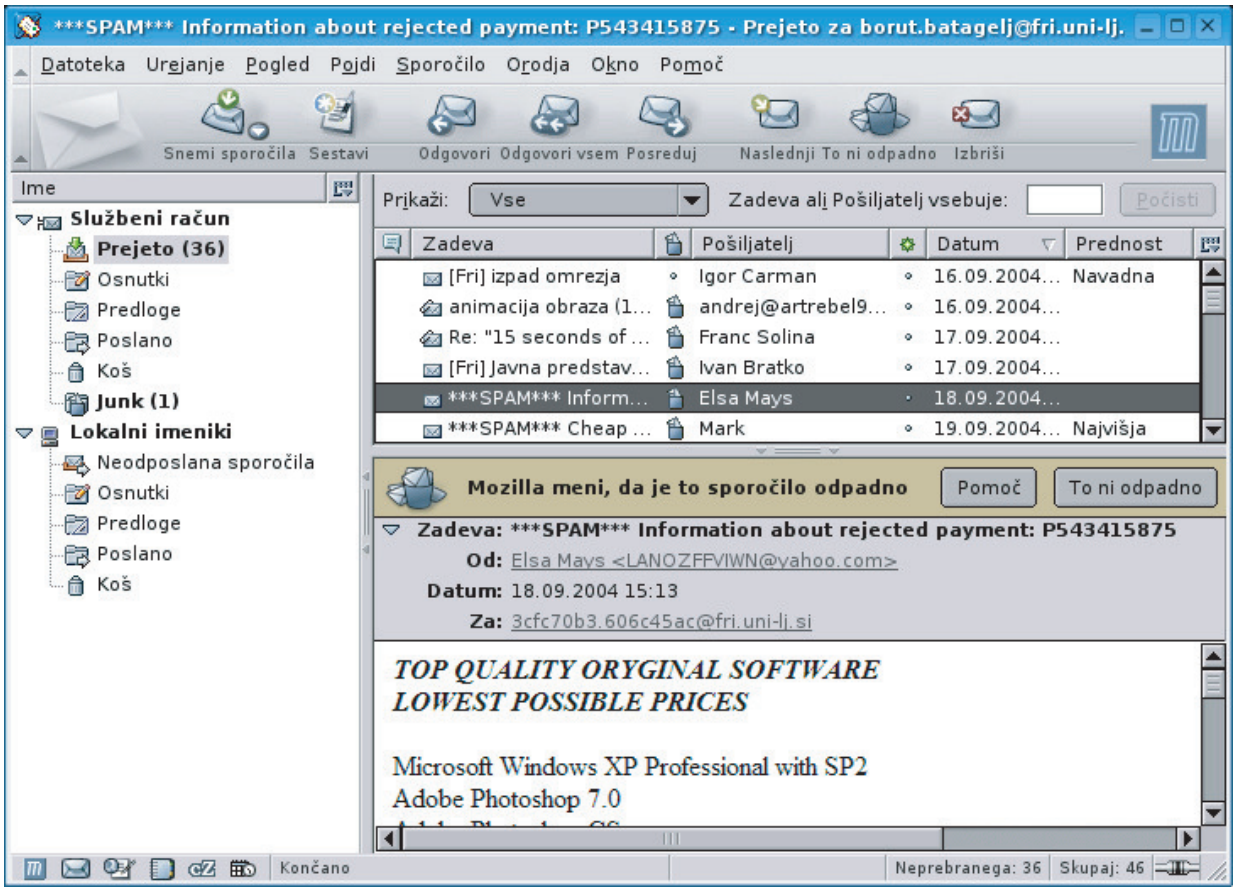
- prikaže seznam z osnovnimi podatki o vseh prispelih sporočilih, ki se nahajajo v uporabnikovem poštnem nabiralniku (angl. *mailbox*),
- omogoča pregledovanje vsebine sporočil in
- sestavljanje ter pošiljanje novih sporočil.

Poseben primer so odjemalci za e-pošto, ki so spletne aplikacije in se izvajajo v brskalniku. Navadno jih uporabljajo javne storitve za elektronsko pošto, kot so Hotmail (`www.hotmail.com`), Yahoo Mail (`www.yahoo.com`) ali Email.si (`www.email.si`).

Če je odjemalec samostojna aplikacija, ki se ne izvaja na istem računalniku kot poštni strežnik, mora poskrbeti tudi za povezavo s strežnikom in prenos sporočil na lokalni računalnik. Poštni strežnik običajno sestavljata dva strežnika: strežnik SMTP poskrbi, da odposlana sporočila prispejo v naslovnikov nabiralnik, strežnik POP3 (ali IMAP) pa za oddaljeni dostop do prispele pošte.

Na sliki 10.5 vidimo samostojni odjemalec e-pošte Mozilla Mail. Omogoča oddaljeni prenos e-pošte preko strežnikov POP3 ali strežnikov IMAP. V nadaljevanju bomo opisali delo z različico, ki je del programske zbirke Mozilla 1.7.2., vendar pa je delo z vsemi samostojnimi odjemalci e-pošte zelo podobno. V operacijskem sistemu Linux ga iz terminalskega okna poženemo z ukazom:
`mozilla -mail &`
Najprej moramo določiti podatke o poštnem strežniku in o svojem računu (angl. *account*) na tem strežniku. Te in še nekatere druge osnovne nastavitve določimo ob prvem zagonu programa ali pa kasneje v kategoriji "Nastavitve računa za Pošto in novičarske skupine ..." menija "Urejanje".

Na levi strani imamo navedene obstoječe račune, s katerih prejemamo e-pošto. Za dodajanje novega računa kliknemo na gumb "Dodaj račun ...". Odpre se pogovorno okno čarovnika, ki nas vodi skozi nastavitve. V prvem pogovornem oknu določimo, da želimo ustvariti nov E-poštni račun. Pod "VaÂše ime:" vpišemo svoje ime in priimek (npr. `Miha Novak`), v polje "E-poštni naslov:" pa elektronski naslov (npr. `miha.novak@podjetje.si`). V naslednjem oknu izberemo tip poštnega strežnika za prihajajočo pošto (npr. POP), v polje "Strežnik za prihajajočo pošto:" vpišemo ime računalnika na katerem se nahaja poštni



Slika 10.5: Odjemalec za e-pošto Mozilla Mail

strežnik za oddaljeni dostop do nabiralnika (npr. `pop.podjetje.si`), v polje "Strežnik za odhodno pošto:" vpišemo ime računalnika, na katerem je nameščen strežnik SMTP, ki ga uporabljamo za pošiljanje pošte (npr. `mail.podjetje.si`). V naslednjem oknu vnesemo v polje "Uporabniško ime:" svoje uporabniško ime na poštnem strežniku (npr. `mihan`). Ker nekateri poštni strežniki SMTP tudi zahtevajo prijavo, moramo uporabniško ime vpisati tudi v polje "Izhodno uporabniško ime:". Običajno sta obe uporabniški imeni enaki. Na koncu še vpišemo poljubno ime za svoj račun (npr. `Domači račun`) in ustvarjanje potrdimo z gumbom "Dokončaj". S pomočjo izbire v meniju "Urejanje→Nastavitve računa za Pošto in novičarske skupine ..." lahko kasneje svoje nastavitve spreminjamo ali pa dodamo nov račun.

Uporabniški vmesnik poštnega odjemalca Mozilla Mail sestavljajo glavni meni, orodna vrstica, trije pomični okvirji in vrstica stanja. V levem zgornjem pomičnem okvirju se nahajajo mape, v katere se shranjujejo sporočila. Pod svojim računom imamo mapo "Prejeto", kjer se nahajajo

sporočila, prenesena s poštnega strežnika. V mapi “Poslano” so kopije uspešno poslanih sporočil, v “Koš” pa sporočila, ki smo jih odstranili iz ene od ostalih map. Če želimo sporočilo dejansko odstraniti iz lokalnega diska, jih moramo torej pobrisati tudi v mapi “Koš”.

Pod glavno mapo “Lokalni imeniki” se v mapi “Neodposlana sporočila” nahajajo sporočila, ki jih odjemalec ni uspel poslati (na primer zaradi prekinjene povezave s strežnikom SMTP). Poštni program jih bo poslal samodejno takoj, ko bo to spet možno. V mapi “Osnutki” se nahajajo sporočila, ki smo jih sestavili in shranili, ne pa tudi odposlali.

V desnem zgornjem okvirju je seznam z osnovnimi podatki o sporočilih, ki se nahajajo v trenutno izbrani mapi, v spodnjem okvirju pa se nahaja vsebina trenutno izbranega sporočila v seznamu. Vsebino sporočila lahko z dvojnim klikom odpremo tudi v novem oknu.

Prenos sporočil iz poštnega strežnika na lokalni disk sprožimo z gumbom iz orodne vrstice “Snemi sporočila”. Ko pravilno vpišemo osebno geslo, se v mapo “Prejeto” prenesejo tista sporočila, ki še niso bila prenesena. Na sporočilo odgovorimo tako, da ga označimo v seznamu in izberemo ikono “Odgovori” iz orodne vrstice. V tem primeru odgovorimo le pošiljatelju. Če želimo, da odgovor prejmejo tudi vsi ostali prejemniki prvotnega sporočila, izberemo ikono “Odgovori vsem”. Če želimo prejeto sporočilo posredovati naprej, izberemo “Posreduj”. Izbrano sporočilo lahko tudi natisnemo ali izbrišemo. Obrazec za sestavljanje novega sporočila prikažemo z gumbom “Sestavi”. Prejeta sporočila imajo lahko pripete datoteke (na primer slike, zvočne zapise, dokumente ipd). Priponke vidimo na dnu sporočila, če v meniju označimo “Pogled→Prikaži priloge kot del sporočila”. Vsebino izbrane priponke lahko pogledamo z dvojnim klikom ali pa shranimo z izbiro “Shrani kot→” iz kontekstnega menija.

Ko sestavljamo novo sporočilo, posredujemo dobljeno sporočilo tretji osebi ali na prejeto sporočilo odgovarjamo, se prikaže pogovorno okno z obrazcem za sestavljenje sporočila (v zadnjih dveh primerih so nekatera polja že izpolnjena). Najprej v polje “Za:” vpišemo elektronski naslov prejemnika sporočila. Kadar je prejemnikov več, ločimo njihove naslove z vejico (npr. `janez.novak@podjetje.si`, `mojca.petek@company.com`). V polje “Zadeva:” vnesemo naslov sporočila. Spodnje, največje vnosno polje, je preprost urejevalnik besedila, v katerega vpišemo vsebino sporočila. Če želimo sporočilu pripeti še datoteke iz lokalnega diska, kliknemo na gumbi “Priloži” iz orodne vrstice pogovornega okna. Prikaže se pogovorno okno, v katerem izberemo datoteke.

10.3.4 Nezaželena elektronska pošta

Pošiljanje nezaželenih oglasnih sporočil (angl. *spam*) je v nekaterih državah z zakonodajo že prepovedano. Med njimi je tudi Slovenija. To

pomeni, da si lahko zaradi nepravilnega pošiljanja elektronskih sporočil kaznovan z denarno kaznijo 3 miljone SIT. Problem pa je, da je še vedno veliko držav, ki nimajo tega zakonsko urejeno in ravno iz teh držav dobimo največ nezaželene elektronske pošte. Po nekaterih raziskavah zaseda nenaročena pošta ob koncu leta 2003, že več kot 60% celotnega prometa elektronske pošte.

Elektronsko sporočilo lahko označimo kot spam, kadar:

1. sta indentiteta prejemnika in vsebina sporočila neveljavna ali prikrita in
2. prejemnik ni preverljivo dovolil pošiljanja sporočila na njegov naslov in tega dovoljenja ne more na enostaven način preklicati in
3. pošiljatelj ima neprimerno večjo korist od sporočila kot prejemnik ali
4. je glava sporočila lažna, ker je pošiljatelj želel prikriti svojo identiteto.

Poznamo dva načina, ki urejata pošiljanje nezaželene elektronske pošte: *opt-in* in *opt-out*. Opt-in označuje metodo, pri kateri uporabnik vnaprej privoli v prejemanje ponudb ali oglasov. Pri metodi opt-out pa se lahko prejemnik odloči, da ne želi sporočila šele, ko je to že prejel. Razlika med prvo in drugo metodo je bistvenega pomena, saj opt-out načelo ne zagotavlja zadostne zaščite pred pošiljanjem spama.

Poleg poslovne škode postaja nenaročena oglasna pošta prava nadloga tudi pri osebni uporabi elektronske pošte. Večina uporabnikov se še vedno povezuje v internet s klicnim dostopom, tako da morajo za nenaročeno pošto poleg vsega še plačevati impulze in naročnino.

Odjemalec elektronske pošte Mozilla Mail ima že vgrajen mehanizem za prepoznavo nezaželene pošte (slika 10.5). V začetku moramo v odjemalcu ročno označiti reklamna sporočila in ga s tem "naučiti", da bo naslednjič lahko na podlagi ključnih besed sam razvrstil nezaželeno sporočilo. V meniju "Orodja→Nadzor odpadne pošte ..." lahko izberemo, ali se bodo nezaželena sporočila zbrisala ali pa prenesla v posebno mapo. Priporočljivo je občasno pregledati tudi mapo z nezaželeno pošto, če ni bilo mogoče katero sporočilo po pomoti označeno kot reklamno.

Dandanes pa ima že veliko poštnih strežnikov vgrajene tako imenovane anti-spam filtre, ki filtrirajo nezaželeno pošto že na strani strežnika. Nekateri strežniki nezaželeno sporočilo samo označijo, drugi pa jo premaknejo v posebno mapo. V drugem primeru se tako rešimo tudi težave prenosa nepomembnih sporočil iz strežnika na lokalni računalnik, priporočljivo pa je, da na strežniku vseeno občasno pregledamo tudi mapo z nezaželeno pošto.

10.4 Jezik HTML

HTML (angl. *HyperText Markup Language*) je označevalni jezik, ki opisuje oblikovne lastnosti spletne strani. Spletna stran (angl. *web page*) je običajna tekstovna datoteka, ki vsebuje hipertekst – besedilo z vsebino strani, opremljeno z ukazi HTML (angl. *HTML tags*). Označbe HTML so preprosti ukazi, s katerimi opišemo strukturo spletnih strani. Način označevanja je podoben kot pri urejanju besedila z LaTeX-om. Osnovne oznake HTML so namenjene definiranju strukture spletnih dokumentov, kasneje dodane oznake pa omogočajo tudi oblikovanje, na primer določanju vrste in velikosti pisave ipd. [8]. Na spletno stran lahko gledamo tudi kot na navodilo spletnemu brskalniku za prikaz določene vsebine.

V tem poglavju se bomo osredotočili na izdelavo preprostih spletnih strani z osnovnimi ukazi HTML. Spletne strani lahko izdelujemo na različne načine. Obstaja veliko orodij, predvsem takih, ki omogočajo interaktivno in vizualno oblikovanje, vendar hkrati skrijejo principe delovanja spletnih strani ter jezika HTML. Ker je naš namen prav v razumevanju HTML, jih ne bomo uporabljali.

Za preizkušanje primerov v nadaljevanju potrebujemo le navaden ASCII urejevalnik besedila (na primer `emacs`) in spletni brskalnik. Z urejevalnikom besedila napišemo spletno stran in jo shranimo na lokalni disk kot običajno tekstovno datoteko s končnico `html`, na primer:
`/home/users/upo/JanezNovak/vaja1.html`
Odpremo jo z brskalnikom iz menija ("Datoteka→ Odpri datoteko ..." v brskalniku Mozilla).

10.4.1 Označbe HTML

Označbe HTML se od vsebine strani ločijo po tem, da se nahajajo med znakoma `<` in `>`. Večinoma nastopajo v parih – vsaka začetna označba se konča s pripadajočo zaključno označbo, ki se od začetne loči po znaku `/`, ki stoji pred imenom oznake (na primer `<HTML>` in `</HTML>`). Označbe so običajno gnezdene, kar pomeni, da med paroma označb nastopajo nove označbe. Označbe imajo lahko tudi atribute, ki se nahajajo med označbo ukaza in znakom `>`. Vrednosti atributov običajno pišemo med enojna ali dvojna narekovaja (na primer `<BODY BACKGROUND="#000000">`).

Posebni označbi sta `<!--` in `-->`, ki oklepata komentarje.

10.4.2 Osnovni gradniki – `HTML`, `HEAD`, `BODY`

Osnovni gradniki spletne strani so:

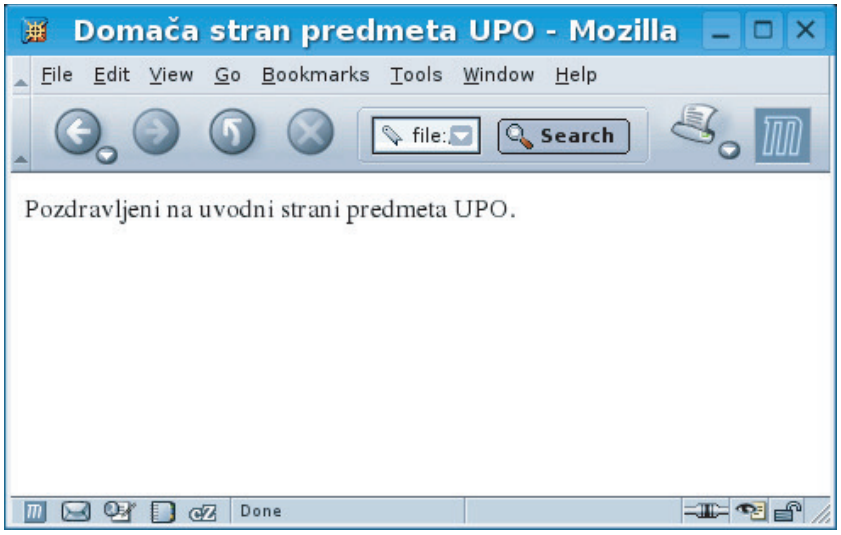
`<HTML> ... </HTML>`	začetek in konec spletne strani
`<HEAD> ... </HEAD>`	glava strani
`<TITLE> ... </TITLE>`	naslov strani
`<BODY> ... </BODY>`	telo strani

Preprosta spletna stran se začne z označbo `<HTML>`, na koncu strani pa je pripadajoča zaključna označba `</HTML>`. Sestavljena je iz glave (`<HEAD>` in `</HEAD>`) in telesa (`<BODY>` in `</BODY>`) dokumenta, ki sta vgnezdena med ustrezni označbi. V glavo dokumenta lahko z ukazom `TITLE` dodamo naslov spletne strani, ki se prikaže v naslovni vrstici spletnega brskalnika. Vsebino spletne strani, ki se bo prikazala v oknu brskalnika, vpišemo med označbi `<BODY>`. `BODY` podpira veliko število atributov za oblikovanje spletne strani, ki si jih bomo ogledali v nadaljevanju.

Primer preproste spletne strani, zgrajene iz osnovnih gradnikov:

```
<HTML>
<HEAD>
<TITLE>Domača stran predmeta UPO</TITLE>
</HEAD>
<BODY>
Pozdravljeni na uvodni strani predmeta UPO.
</BODY>
</HTML>
```

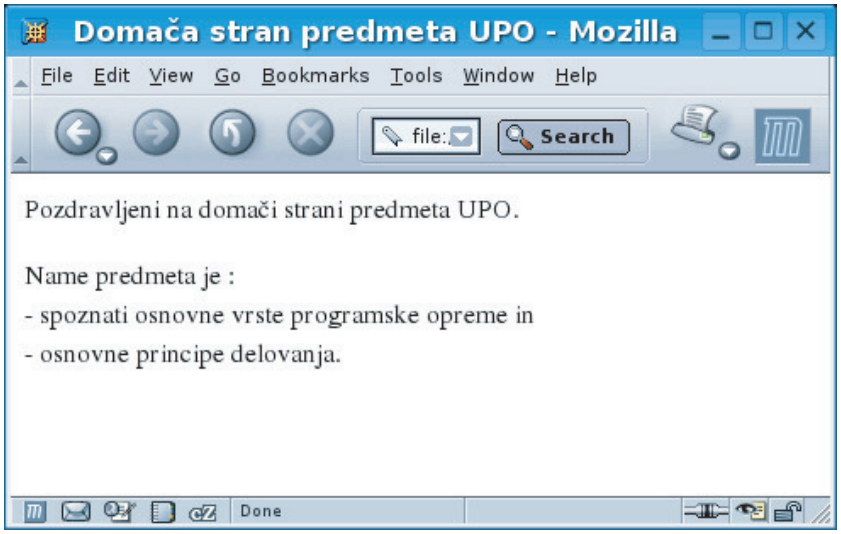
Izgled primera v brskalniku je prikazan na sliki 10.6.



Slika 10.6: Preprosta spletna stran v brskalniku

10.4.3 Odstavki, presledki in nove vrstice

Beseda v HTML je enako kot v vseh naravnih jezikih niz zaporednih znakov, ki so od ostalega besedila ločeni z vsaj enim presledkom. Število presledkov med njimi ni pomembno. Brskalnik bo stran prikazal enako ne glede na število presledkov med besedami.



Slika 10.7: Primer ločevanja besed in odstavkov

Za oblikovanje odstavkov uporabljamo ukaza:

` `	prelom vrstice
`<P>`	nov odstavek

Odstavke ločimo z označbo `<P>`. Brskalnik pri prikazu ignorira vse odvečne prazne vrstice in presledke v kodi HTML. Če želimo del besedila izpisati v novi vrstici, a v istem odstavku, uporabimo označbo `<BR>`. Pri označbah `<P>` in `<BR>` lahko izpustimo pripadajoče končne označbe.

Za razdelitev HTML dokumenta na bloke lahko uporabimo tudi oznako `<DIV>`. Ukazi `DIV` so lahko tudi gnezdeni. Pravo uporabnost pa dobijo šele v kombinaciji s slogovnimi predlogami CSS (Cascading Style Sheets). Podobno vlogo ima tudi ukaz `SPAN`, ki pa je namenjen oblikovanju posamezne vrstice.

Ločevanje besed in odstavkov je prikazano v naslednjem primeru (slika 10.7):

```
<HTML>
<HEAD>
<TITLE>Domača stran predmeta UPO</TITLE>
</HEAD>
<BODY>
```

```
Pozdravljeni    na domači


strani predmeta UPO. <P>

Namen predmeta je:  <BR>
- spoznati osnovne vrste programske opreme in <BR> - osnovne
principe delovanja.
</BODY> </HTML>
```

10.4.4 Šumniki

Če želimo, da brskalnik prikaže tudi šumnike, moramo označiti, v kateremu naboru znakov (angl. *code page*) smo izdelali izvorno datoteko HTML. To storimo s posebnim ukazom v glavi strani:

```
<META HTTP-EQUIV="Content-Type" content="text/html;
charset=iso-8859-2">
```

Primer prikaza strani v "ISO Latin 2" kodnemu naboru (slika 10.8):

```
<HTML>

<HEAD>
<META HTTP-EQUIV="Content-Type" content="text/html;
charset=iso-8859-2">
<TITLE>Šumniki v naslovu</TITLE>
</HEAD>
<BODY>
Šumniki niso več težava.
</BODY>
</HTML>
```

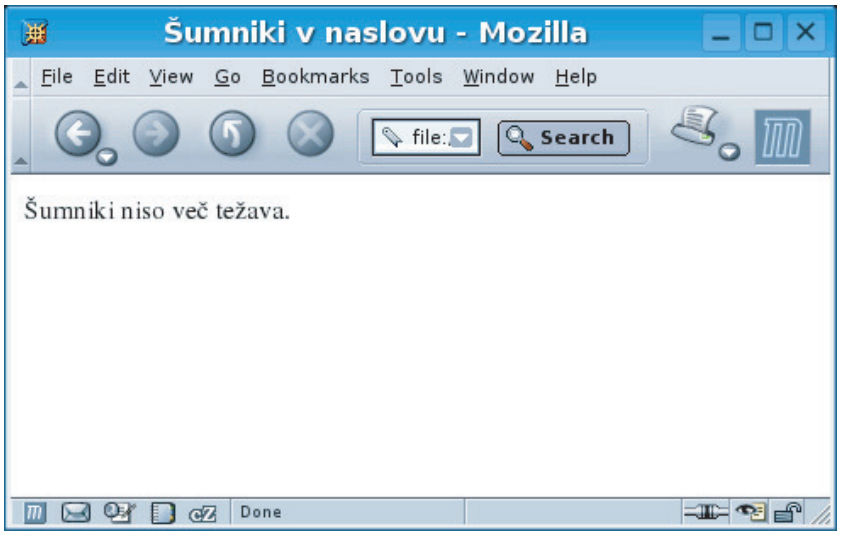
V vseh primerih, ki sledijo v nadaljevanju, bomo privzeli takšno glavo strani HTML, ki omogoča prikaz šumnikov. Zato bomo navajali le osrednji del strani v označbi `<BODY>`.

10.4.5 Naslovi

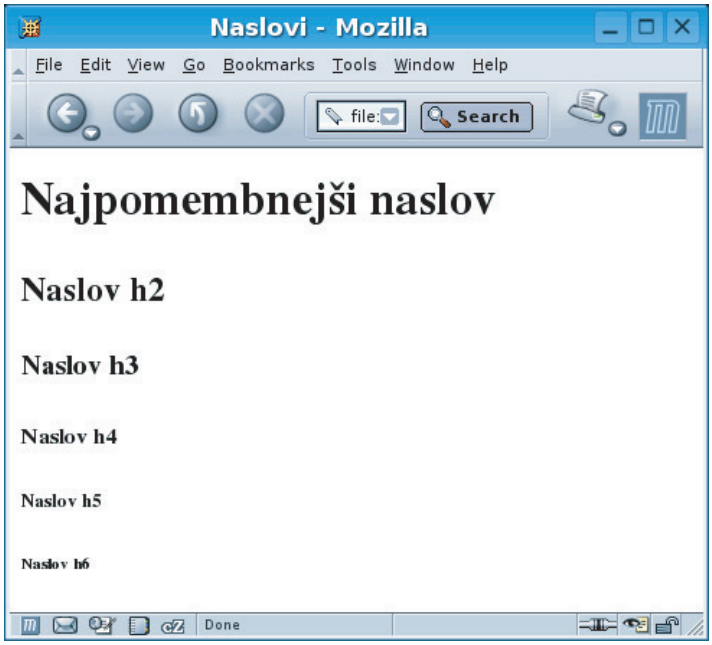
HTML pozna šest nivojev naslovov, ki jih označujemo na naslednji način:

`<Hn> Naslov </Hn>`; n=1 ... 6.
Naslov `H1` je najbolj dominanten in se prikaže z največjo pisavo. Vsi naslovi so običajno prikazani poudarjeno in z večjo pisavo kot običajno besedilo.

Primer (slika 10.9):



Slika 10.8: Prikaza šumnikov v brskalniku



Slika 10.9: Primer različnih naslovov v brskalniku

```
<BODY>
<H1>Najpomembnejši naslov</H1>
<H2>Naslov h2</H2>
<H3>Naslov h3</H3>
<H4>Naslov h4</H4>
<H5>Naslov h5</H5>
<H6>Naslov h6</H6>
```

```
</BODY> .
```

10.4.6 Hipertekstovne povezave

Ena pomembnih lastnosti spletnih strani je, da lahko uporabljamo hipertekstovne povezave. Ločimo dve vrsti povezav:

- povezave z drugimi spletnimi stranmi in
- povezave v sklopu iste strani.

Povezavo na drugo spletno stran vstavimo z ukazom:

```
<A HREF="URL"> naziv povezave </A>
```

`URL` (angl. *Uniform Resource Locator*) je enolična oznaka spletne strani, ki je opisana v razdelku 10.3.1. Primeri protokolov, ki se lahko uporabljajo v oznaki URL, so: `file` (dostop do datotek na lokalnem računalniku), `http` (dostop do datotek na spletnem strežniku), `ftp` (dostop do datotek na datotečnem strežniku), `telnet` (dostop do oddaljenega računalnika) in `mailto` (pošiljanje elektronske pošte).

Povezave v okviru istega dokumenta uporabljamo za premike po isti spletni strani. Smiselne so predvsem takrat, ko je spletna stran dolga, bralca pa utegnejo zanimati le določeni deli te spletne strani. Ta mesta zaznamujemo z enoličnimi oznakami:

```
<A NAME="oznaka"> .
```

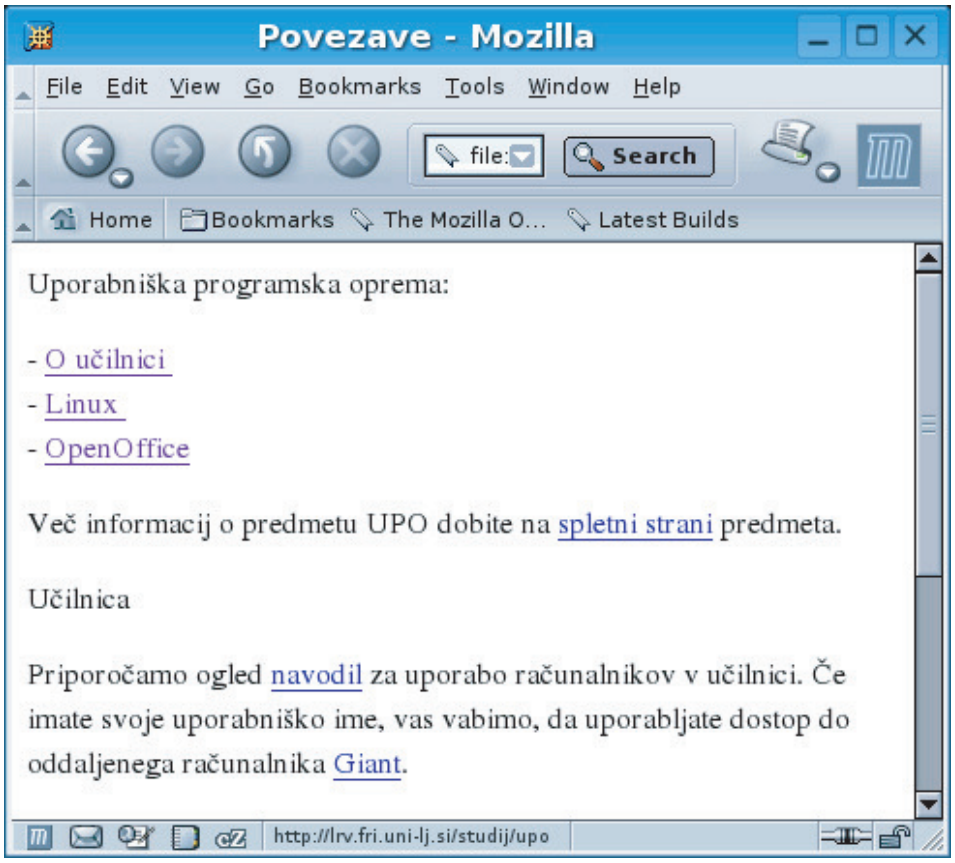
Povezavo z označenim mestom v besedilu vstavimo podobno kot povezavo z drugo spletno stranjo, le da se namesto na URL sklicujemo na oznako na isti spletni strani:

```
<A HREF="#oznaka"> povezava </A> .
```

Primer spletne strani, ki prikazuje uporabo različnih povezav (slika 10.10):

```
<BODY>
Uporabniška programska oprema:  <P>

- <A HREF="#ucilnica"> O učilnici </A> <BR>
- <A HREF="#linux"> Linux </A> <BR>
- <A HREF="#office"> OpenOffice </A> <P>

Več informacij o predmetu UPO dobite na
<A HREF="http://lrv.fri.uni-lj.si/studij/upo">spletni
strani</A> predmeta.  <P>

<A NAME="ucilnica">
```

```
Učilnica <P>
Priporočamo ogled <A HREF="file:/users/upo/navodila.txt">
navodil</A> za uporabo računalnikov v učilnici.  Če imate
svoje uporabniško ime, vas vabimo, da uporabljate dostop do
oddaljenega računalnika
<A HREF="telnet://giant.fri.uni-lj.si">Giant</A>.  <P>

<A NAME="linux">
Linux <P>
Na vajah bomo najprej spoznali operacijski sistem Linux.<P>

<A NAME="office">
OpenOffice <P>
Večji del vaj bo namenjen paketu OpenOffice.<P>

Pripombe in predloge pošljite
<A HREF="mailto:urednikUPO@domena.si"> uredniku</A>.
</BODY>
```



Slika 10.10: Izgled primera z različnimi povezavami v brskalniku

10.4.7 Oblikovanje pisave

Ukazi za oblikovanje pisave natančneje določajo videz dela besedila na zaslonu. Barvo in velikost pisave določimo z lastnostima ukaza `<FONT>`:

```
<FONT SIZE="+|-Y"> ...  </FONT>
<FONT SIZE="Y"> ...  </FONT>
<FONT COLOR="#RRGGBB"> ...  </FONT>
```

Z lastnostjo `SIZE` v prvem primeru relativno povečamo ali zmanjšamo velikost pisave glede na privzeto velikost, v drugem pa jo določimo absolutno. Lastnost `COLOR` določa vrednost RGB barve besedila. Barva besedila je predstavljena s kombinacijo treh osnovnih barv: prvi mesti določata prisotnost rdeče, drugi zelene in zadnji modre barve. Vsaka osnovna barva je določena z vrednostjo šestnajstiškega števila na območju od 00 do FF. Na primer vrednost RGB črne barve je 000000 (ne vsebuje nobene barve), bele pa FFFFFF (vse barve so v nasičenju).

Oblikovanja pisave z ukazom `FONT`:

```
<BODY>
<FONT SIZE="+1"> Za 1 večja pisava.  </FONT>
<FONT SIZE="-3"> Za 3 manjša pisava.  </FONT>
<FONT SIZE="5"> Pisava velikosti 5.  </FONT>
</BODY>
```

Z naslednjimi ukazi pa lahko izbiramo različne sloge pisav:

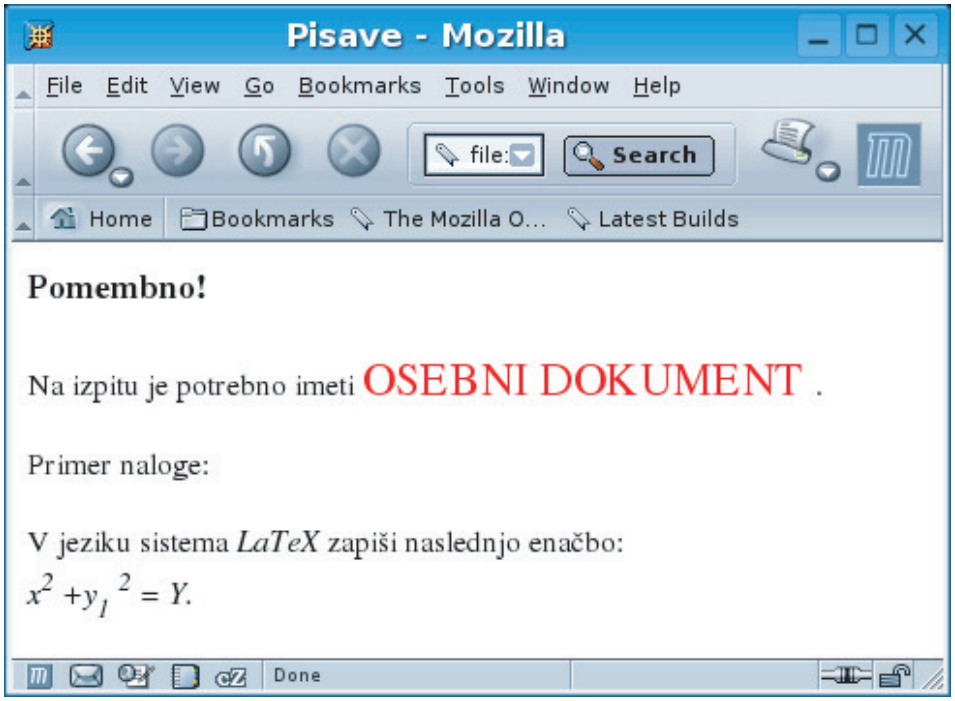
`<B> krepko </B>`	**krepko** besedilo
`<I> kurzivno </I>`	*kurzivno* besedilo
`<B><I> kombinirano </I></B>`	***kombinirano*** oblikovanje
`<TT> neproporcionalno </TT>`	`neproporcionalna` pisava
`<U> podčrtano </U>`	podčrtano besedilo
`<BLINK> utripajoče </BLINK>`	utripajoče besedilo
`<SUP> potenca </SUP>`	dvignjeno besed. (angl. *superscript*)
`<SUB> indeks </SUB>`	spuščeno besedilo (angl. *subscript*)

Besedilo lahko oblikujemo tudi s tako imenovanimi "logičnimi" stili. V tem primeru s posebnimi ukazi označimo dele besedila, ki imajo poseben pomen. Prikaz besedila ni natančno določen s strani oblikovalca, ampak je odvisen tudi od nastavitev brskalnika.

`<BIG> velike </BIG>`	velike črke
`<SMALL> majhne </SMALL>`	majhne črke
`<STRONG> poudarjeno </STRONG>`	poudarjeno besedilo
`<CITE> citat </CITE>`	označevanje citatov
`<CODE> koda </CODE>`	označevanje kode
`<SAMP> primer </SAMP>`	označevanje primerov

Primer oblikovanja pisave (slika 10.11):

```
<BODY>
<BLINK><B> <BIG>Pomembno!  </BIG> </B></BLINK> <P>

Na izpitu je potrebno imeti
<FONT SIZE="5" COLOR="#FF0000"> OSEBNI DOKUMENT </FONT>.<P>

Primer naloge:  <P>
V jeziku sistema <I>LaTeX</I> zapiši naslednjo enačbo:  <BR>
<CITE>
x<SUP>2</SUP> + y<SUB>1</SUB><SUP>2</SUP> = Y.
</CITE>
</BODY>
```



Slika 10.11: Izgled primera z različnimi oblikami pisave

Nobena od naštetih označb ne upošteva oblikovanja besedila v izvorni datoteki HTML. Če želimo, da se pri prikazu besedila dosledno upoštevajo zaporedni presledki, tabulatorji in prelomi vrstic, lahko uporabimo ukaz `PRE`. Gre za posebni logični stil, ki označuje vnaprej oblikovano (angl. *preformated*) besedilo, podobno kot v okolju `verbatim` v LaTeX-u.

Primer vnaprej oblikovanega besedila (slika 10.12):

```
<BODY>
Rezultati zagovora vaj:<P>
```

```
<PRE>
Janez Novak             10
Miran A. Rojnik         9
Ana Gril                9
Miha Moder              -
</PRE>
</BODY>
```

Slika 10.12: Izgled primera vnaprej oblikovanega besedila

Določeni znaki se ne nahajajajo v običajni tabeli ASCII (na primer znak za avtorske pravice) ali pa imajo v HTML poseben pomen (<, > in &).

` `	nedeljiv presledek
`<`	< (manjši kot)
`>`	> (večji kot)
`&`	& (in)
`"`	"(dvojni narekovaj)
`ñ`	ñ (n s tildo)
`É`	É (E z ostrivcem)
`ä`	ä (a s preglasom)
`©`	©
`÷`	znak s kodo ASCII 247

10.4.8 Oblikovanje odstavkov

Za oblikovanje poravnave odstavkov uporabljamo lastnost `ALIGN` ukazov `P`, `DIV` in `Hn` (n=1,..,6):

```
<DIV ALIGN="LEFT|CENTER|RIGHT"> ...  </DIV>
<P ALIGN="LEFT|CENTER|RIGHT"> ...  </P>
<Hn ALIGN="LEFT|CENTER|RIGHT"> Naslov </Hn>; n = 1, ..., 6
```

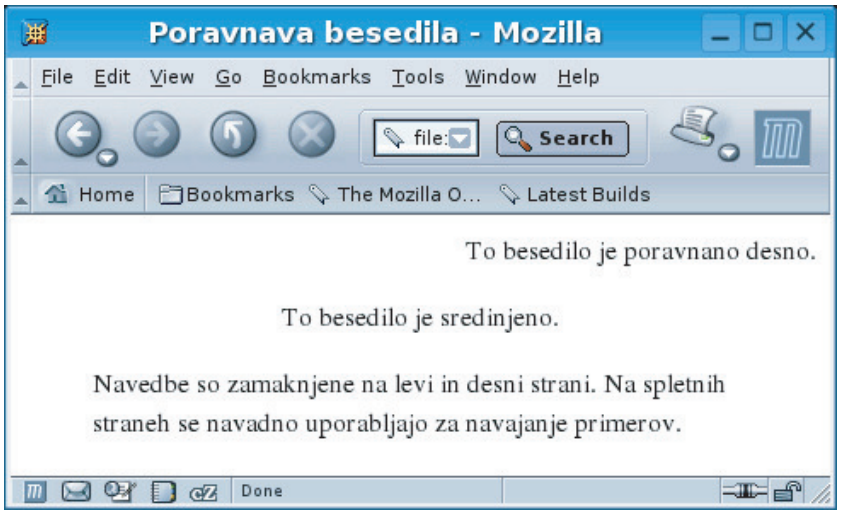
`LEFT` pomeni levo poravnavo, `RIGHT` desno, `CENTER` pa sredinjenje odstavka glede na širino vrstice v brskalniku. Uporaba zaključne označbe je v tem primeru obvezna, saj s tem označimo, kdaj poravnava preneha veljati. Za sredinjenje lahko uporabimo tudi označbi `<CENTER>` in `</CENTER>`.

Poseben primer logičnega oblikovanja odstavkov so bločne navedbe, ki so navadno prikazane z zamikom od levega roba besedila. Uporabljajo se za navajanje primerov ali citatov. Definirata jih označbi:

```
<BLOCKQUOTE> ...  </BLOCKQUOTE>
```

Primer različnih poravnav (slika 10.13):

```
<P ALIGN="RIGHT"> To besedilo je poravnano desno.</P>
<CENTER> To besedilo je sredinjeno.</CENTER>

<BLOCKQUOTE> Navedbe so zamaknjene na levi in desni strani.
Na spletnih straneh se navadno uporabljajo za
navajanje primerov.
</BLOCKQUOTE>
```



Slika 10.13: Izgled primera različnih poravnav besedila

10.4.9 Vodoravne črte

Vodoravna črta služi kot pripomoček za členjenje dokumentov. Vstavimo jo z ukazom:

```
<HR WIDTH="X" ALIGN="LEFT|CENTER|RIGHT" SIZE="Y">
```
Primer:

```
<HR>
<HR ALIGN="CENTER" WIDTH="50%">
<HR ALIGN="LEFT" WIDTH="400" SIZE="10">
```

10.4.10 Seznami

HTML podpira tri vrste seznamov: neurejene in urejene sezname ter sezname definicij. Pri tem lahko poljubno gnezdimo tako istovrstne sezname kot tudi različne sezname med seboj.

Neurejeni seznami

Neurejeni seznam je najenostavnejša oblika zaporednega seznama, pri katerem številčenje ni potrebno. Navadno gre za naštevanja, pri katerih vrstni red ni pomemben. Vstavimo ga z ukazom `UL`:

```
<UL [TYPE="disc|circle|square"]>
<LI>
...
</UL>
```
Z `<LI>` vstavimo v seznam posamezni element. Lastnost `TYPE` določa znak, ki se prikaže pred vsakim elementom.

Urejeni seznami

Urejeni seznam je podoben neurejenemu, le da so posamezni elementi seznama oštevilčeni:

```
<OL [TYPE="a|A|i|I"] [START="N"]>
<LI [VALUE="X"]>
...
</OL>
```
Enako kot pri neurejenem seznamu vstavimo z `<LI>` posamezni element. Lastnost `TYPE` ukaza `OL` določa vrsto številčenja elementov, lastnost `START` pa cifro, s katero se številčenje začenja. To cifro lahko pri posameznem elementu spremenimo z lastnostjo `VALUE`.

Seznami definicij

Seznam definicij sestavljata dve vrsti elementov. Najprej navedemo izraz, ki ga želimo definirati. Sledi besedilo, ki izraz razlaga. Slednje besedilo je navadno prikazano z odmikom od levega roba brskalnika.

```
<DL>
<DT> izraz
<DD> besedilo definicije
...
</DL>
```

Primer spletne strani s seznami (slika 10.14):

```
<BODY>
Znanja potrebna za zagovor vaj:
<OL TYPE="A">
<LI> poznavanje HTML
<LI> poznavanje OS Linux, in sicer:
<UL TYPE="SQUARE">
<LI> dela v grafičnem načinu in
<LI> možnosti v okenskem načinu.
</UL>
<LI> poznavanje paketa OpenOffice.org.
</OL>

<DL>
<DT> HTML
<DD> je označevalni jezik za izdelavo spletnih strani.
</DL>

</BODY>
```

10.4.11 Ozadje dokumenta

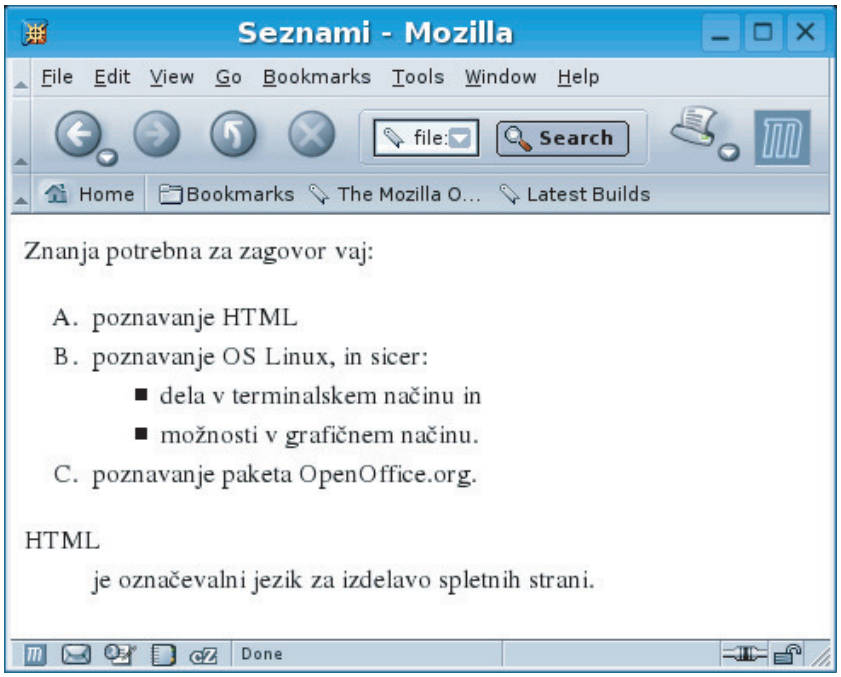
Lastnost `BACKGROUND` oznake `<BODY>` omogoča postavitev slike v ozadje spletne strani.

```
<BODY BACKGROUND="URL">  .
```

Z različnimi lastnostmi oznake `<BODY>` lahko določamo barvo ozadja, besedila in povezav.

```
<BODY BGCOLOR="#RRGGBB" TEXT="#RRGGBB" LINK="#RRGGBB"
VLINK="#RRGGBB" ALINK="#RRGGBB">  .
```

Barvo določimo z šestnajstiškimi vrednostmi posameznih kanalov RGB podobno kot pri ukazu `FONT`. Tako pri vstavljanju slik kot pri spreminjanju barve ozadja pa moramo biti pazljivi, da ostaja besedilo dobro vidno.



Slika 10.14: Primer različnih seznamov

Primer:

```
<BODY BACKGROUND="slike/ozadje.gif"
TEXT="#0000FF" LINK="#00FF00">
Povezava je tokrat <A HREF="http://www.fri.uni-lj.si">
zelene barve</A>.
</BODY>  .
```

10.4.12 Vstavljanje slik

Sliko v formatu GIF, PNG ali JPEG vstavimo z ukazom:

`<IMG SRC="URL">` .
`URL` je oznaka URL slike, ki bi jo radi vstavili, in se v splošnem lahko nahaja kjerkoli v svetovnem spletu. Vendar pa slike običajno pustimo kar v istem imeniku kot spletno stran, v katero jih vstavljamo. V tem primeru je oznaka URL kar ime slike:

`<IMG SRC="ime_slike.gif">` .
Če imamo na svojem spletnem mestu veliko število slik, je smotrno, da jih shranimo v svoj podimenik, ki je rezerviran samo zanje. V tem primeru bo sintaksa:

`<IMG SRC="podimenik/ime_slike.gif">` .

Primer z vključevanjem slik:

```
<BODY>
<IMG SRC=
"http://www.burger.si/TriglavNationalPark/Triglav/VRH_Triglav.jpg">
<P>
<IMG SRC="VRH_Triglav.jpg"> <P>
<IMG SRC=slike/VRH_Triglav.jpg> <P>
</BODY>  .
```

Slika se lahko nahaja v svojem odstavku, lahko pa jo vstavimo v isti odstavek kot spremljajoče besedilo. Kadar oblikujemo besedilo, ki se nahaja v istem odstavku kot slike, si želimo čim večjo prilagodljivost pri pretoku besedila glede na položaj slike. V HTML lahko slike umestimo levo, desno ali sredinjeno glede na besedilo poleg njih in jih z besedilom poravnamo na zgornjem ali spodnjem robu:

`<IMG ALIGN="TOP|BOTTOM|MIDDLE|LEFT|RIGTH" SRC="URL">` .
Po privzeti nastavitvi so slike umeščene v spodnjem levem robu besedila.

Primer (Slika 10.15):

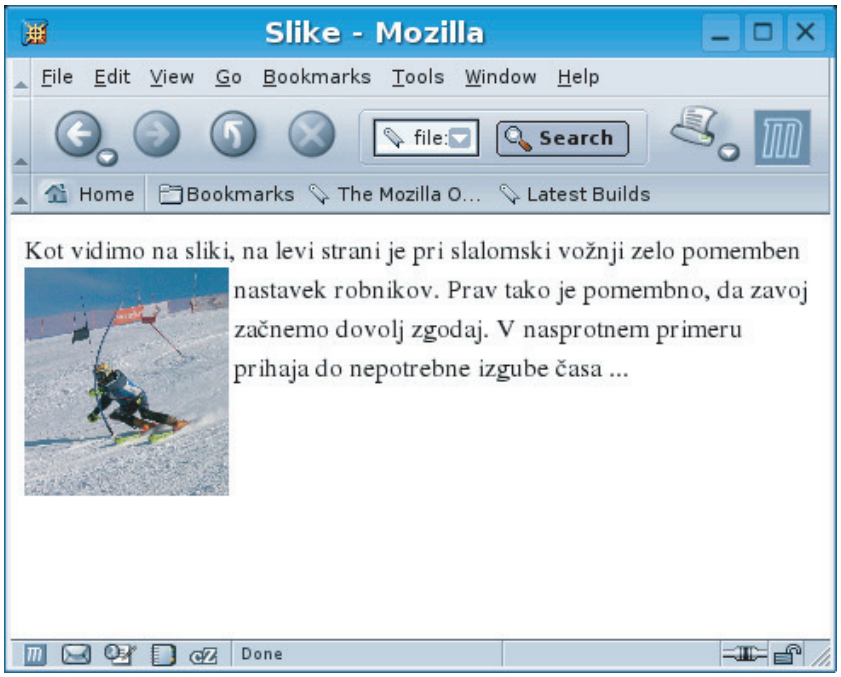
```
<BODY>
Kot vidimo na sliki na levi strani
<IMG ALIGN="LEFT" SRC="image.jpg" WIDTH="120">
je pri slalomski vožnji zelo pomemben nastavek
robnikov.  Prav tako je pomembno, da zavoj začnemo
dovolj zgodaj.  V nasprotnem primeru prihaja do
nepotrebne izgube časa ...<P>
</BODY>  .
```

10.4.13 Slike, občutljive na dotik

V dokumentih HTML so priljubljene slike, ki so občutljive na dotik. Tovrstna slika nas pri kliku nanjo poveže na nov spletni dokument. Povezava je odvisna od mesta v sliki, na katerega smo kliknili. Primeri slike, občutljive na dotik, so zemljevid, turistična karta ipd.

Najprej z ukazom `MAP` določimo območja, ki bodo občutljiva na dotik, in hipertekstne povezave, na katere ta območja kažejo:

```
<MAP NAME="ime_mape">
<AREA SHAPE="rect|circle|poly|default" COORDS="x,y,..."
HREF="URL_povezave">
...
</MAP>
```

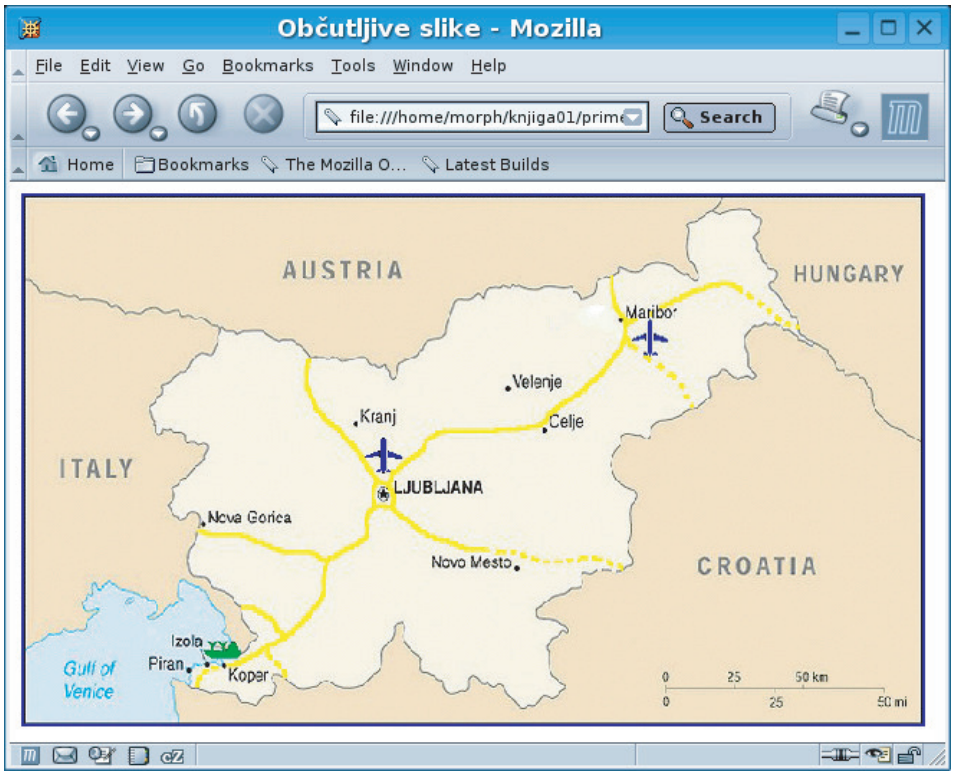


Slika 10.15: Izgled primera s sliko, poravnano na levi strani

Sliko, občutljivo na dotik, vstavimo kot navadno sliko, ki uporablja občutljiva območja, določena z `MAP`:

```
<IMG SRC="URL_slike" USEMAP="ime_mape">
```
.
Primer slike, občutljive na dotik (Slika 10.16):

```
<BODY>
<MAP NAME="SloMesta">
<AREA SHAPE=RECT COORDS="408,84,426,94"
HREF="http://www.maribor.si">
<AREA SHAPE=CIRCLE COORDS="249,212,20"
HREF="http://www.ljubljana.si">
<AREA SHAPE=POLY COORDS="119,260,171,303,118,337"
HREF="http://www.obala.net">
<AREA SHAPE=DEFAULT HREF="http://www.najdi.si">
</MAP>

<IMG SRC="mapa-slo.gif" USEMAP="#SloMesta" WIDTH="625"
HEIGHT="367">
</BODY>
```
.
Atribut `COORDS` predstavlja pri pravokotniku (`RECT`) in večkotniku (`POLY`) koordinate oglišč. Krog (`CIRCLE`) je predstavljen s koordinatami središča in polmerom.



Slika 10.16: Primer slike, občutljive na dotik

10.4.14 Tabele

Tabelo vstavimo v spletni dokument z ukazom `TABLE`. Praviloma je sestavljena iz naslova, vrstic in celic. Zgradbo tabele in njene oblikovne lastnosti določa več različnih označb:

- S `<TABLE>` začnemo tabelo. Obvezno jo moramo zapreti z označbo `</TABLE>`. Lastnost `BORDER="X"` doda okoli vsake celice okvir debeline `X`.
- Ukaz `CAPTION` doda tabeli naslov, ki se nahaja pred tabelo. Z lastnostjo `ALIGN="BOTTOM"` ga prestavimo za tabelo.
- `<TR>` je označba za začetek vrstice. Označbo moramo končati s `</TR>`, sledi pa ji lahko začetek naslednje vrstice ali zaključek tabele.
- `<TH>` in `</TH>` sta označbi za celico v naslovni vrstici. Od označbe `<TD>` se razlikujeta po tem, da je besedilo samodejno sredinjeno in poudarjeno.
- `<TD>` in `</TD>` sta označbi za običajno celico.

Označbe <TR>, <TH> in <TD> lahko vsebujejo tudi lastnosti za poravnavo besedila:

- ALIGN="LEFT|RIGHT|CENTER" določa vodoravno poravnavo celice ali vrstice.
- VALIGN="TOP|MIDDLE|BOTTOM" določa navpično poravnavo celice ali vrstice.
- COLSPAN="X" – določa število stolpcev, čez katere se razteza celica (širina celice).
- ROWSPAN="Y" – določa število vrstic, čez katere se razteza celica (višina celice).
- NOWRAP - izključi prelom besedila glede na širino celice.

Seveda uporaba vseh lastnosti posameznih označb ni obvezna. Najpreprostejša definicija tabele je zato kar:

```
<TABLE>
<TR> <TD> ...  </TD> ...  </TR>
...
</TABLE>
```

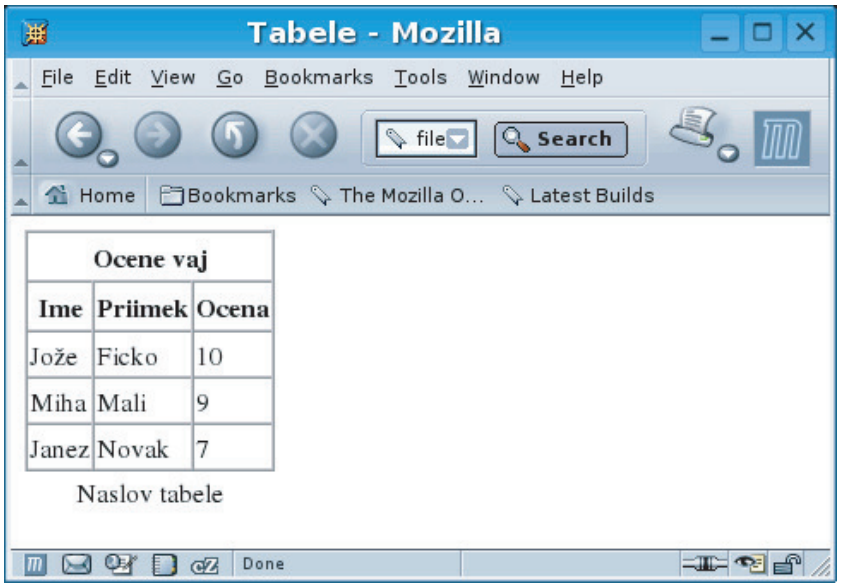
Popolnejša definicija tabele pa je:

```
<TABLE [BORDER="X"] [CELLSPACING="Y"] [CELLPADDING="Z"]
[WIDTH="W"]>
<CAPTION [ALIGN="TOP|BOTTOM"]> Naslov tabele </CAPTION>
<TR [ALIGN=" "] [VALIGN=" "]>
<TD [ALIGN=" "] [VALIGN=" "] [NOWRAP] [COLSPAN=" "]
[ROWSPAN=" "] [WIDTH=" "]>
</TD>
...
</TR>
...
</TABLE>
```

V pomen vseh lastnosti označb se na tem mestu ne bomo spuščali. Bralec jih bo našel v literaturi.

Primer, ki je prikazan na sliki 10.17:

```
<BODY>
<TABLE BORDER="1" CELLSPACING="0" CELLPADDING="1">
<CAPTION ALIGN="BOTTOM"> Naslov tabele </CAPTION>
<TR><TH colspan="3">Ocene vaj</TH> </TR>
<TR><TH>Ime</TH> <TH>Priimek</TH> <TH>Ocena</TH> </TR>
<TR><TD>Jože</TD> <TD>Ficko</TD> <TD>10</TD></TR>
```



Slika 10.17: Primer tabele v HTML

```
<TR><TD>Miha</TD> <TD>Mali</TD> <TD>9</TD></TR>
<TR><TD>Janez</TD> <TD>Novak</TD> <TD>7</TD></TR>
</TABLE>
</BODY>   .
```

10.4.15 Okvirji

Okvirji omogočajo razdelitev vidnega področja spletne strani na okvirje (podpodročja). Vsak okvir lahko prikaže vsebino hipertekstnega dokumenta, neodvisno od vsebine ostalih okvirjev. Velikost okvirja se spreminja s spreminjanjem velikosti vidnega področja v brskalniku. Spreminjanje velikosti okvirja lahko dovolimo tudi uporabniku.

Struktura spletnega dokumenta z okvirji je podobna strukturi navadnega hiperteksta. Označbi `<BODY>` in `</BODY>` je potrebno zamenjati s parom `<FRAMESET>`, `</FRAMESET>`, ki vsebuje opis okvirjev hiperteksta:

```
<HEAD>
</HEAD>
<FRAMESET>
</FRAMESET>  .
```

Vsebino hiperteksta z več okvirji določimo z ukazoma `FRAMESET` in `FRAME`. `FRAMESET` določa zgradbo celotnega hiperteksta, `FRAME` pa lastnosti posameznega okvirja. Vsebino okvirjev določimo z novim spletnim dokumentom:

```
<FRAMESET [COLS="C1,C2,..."] [ROWS="R1,R2,..."]>
<FRAME SRC="URL" NAME="ime" [MARGINWIDTH=x] [MARGINHEIGHT=y]
[NORESIZE]>
...
</FRAMESET> .
```

Lastnostima `COLS` in `ROWS` podamo širino (`Ci`) in višino (`Ri`) okvirjev najpogosteje na tri načine:

- `Ri = število`; s številko podamo širino in višino okvirja v točkah. Ker se višina in širina celotnega okna v spletnem brskalniku spreminjata, bo brskalnik sam spremenil velikosti oken, tako da bo celotna površina v brskalniku zasedena.
- `Ri = število%`; število predstavlja delež širine (višine), ki ga okno zaseda v brskalniku. Če je vsota vseh deležev manjša od 100, bo del okna v brskalniku nezaseden. Če je vsota večja kot 100, brskalnik v razmerju zmanjša vse deleže.
- `Ri = *`; okvir bo zavzel preostanek prostora.

Če spletni brskalnik ne zna prikazovati okvirjev, lahko z označbo `<NOFRAMES>` prikažemo sporočilo:

```
<NOFRAMES>
Tu pustimo sporočilo za brskalnik, ki ne zna
prikazovati okvirjev.
</NOFRAMES> .
```

Pri hipertekstnih povezavah lahko pri spletnih dokumentih z večimi okvirji določimo tudi njihov ciljni okvir:

```
<A HREF="URL" TARGET="ime_okvirja"> ... </A>.
```

Za potrebe spletnega dokumenta z več okvirji pripravimo najprej datoteki `lokacija1.html` in `lokacija2.html`, v katerih je vsebina, ki se bo prikazovala v enem od okvirjev. Drugi okvir bo vseboval spletni dokument s kazalom na omenjeni datoteki (`kazalo.html`):

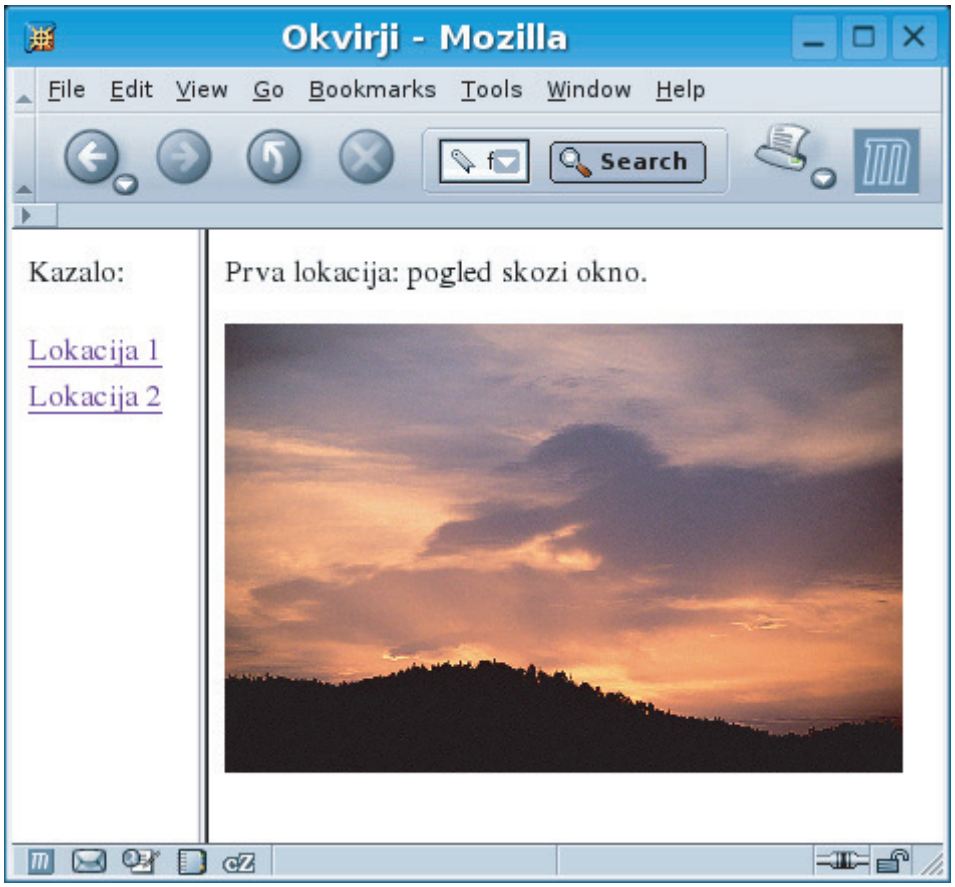
```
<BODY>
Kazalo: <P>
<A HREF="lokacija1.html" TARGET="vsebina"> Lokacija 1 </A>
<BR>
<A HREF="lokacija2.html" TARGET="vsebina"> Lokacija 2 </A>
</BODY>
```

Okvirje povežemo s krovnim spletnim dokumentom, ki ga naložimo v brskalnik:

```
<TITLE> Okvirji </TITLE>
<FRAMESET COLS="20%,80%">
```

```
<FRAME SRC="kazalo.html" NAME="kazalo">
<FRAME SRC="lokacija1.html" NAME="vsebina">
</FRAMESET>
<NOFRAMES>
Vaš brskalnik ne zna prikazovati okvirjev.
</NOFRAMES>
```

Videz gornjega primera v brskalniku je prikazan na sliki 10.18.



Slika 10.18: Spletni dokument z večimi okvirji

10.5 Jezik XML

Jezik XML (angl. *eXtensible Markup Language*, razširljiv označevalni jezik) je standardni označevalni jezik za opis dokumentov, ki vsebujejo strukturirano vsebino in omogočajo še doslednejše ločevanje vsebine od oblike [9]. Beseda "dokument" se nanaša na opis najrazličnejših vsebin v elektronski obliki, kot so na primer preglednice, transakcije elektronskega poslovanja, stanja programskih objektov, protokolna sporočila, vektorska

grafika, nastavitvene datoteke ipd. Namenski programi, ki obdelujejo tovrstne podatke, jih najpogosteje shranjujejo na trdi disk v obliki binarnih datotek (npr. dokument urejevalnika OpenOffice.org Writer). Na XML lahko gledamo kot na sklop smernic in predpisov za generiranje tovrstnih dokumentov v tekstovni obliki. Standard je razširljiv, platformsko neodvisen in podpira internacionalizacijo (različne kodne strani).

Razvoj XML se je začel leta 1996, prva različica pa je bila sprejeta in standardizirana s strani spletnega konzorcija W3C leta 1998. Vendar pa so ideje, ki so vpete v XML, starejše. Pred njim sta obstajala že SGML, standard ISO, razvit v zgodnjih 80-ih, in HTML, katerega razvoj se je začel leta 1990. Obogateni z izkušnjami iz HTML so razvijalci uporabili najboljše dele SGML in razvili standard, ki je prav tako močan kot SGML, a precej enostavnejši.

10.5.1 XML in HTML

Podobno kot pri HTML so tudi pri XML osnovni gradniki dokumenta označbe, tj. besede, ki jih oklepata ločili "<" in ">", ter njihovi atributi. Na primer označba `AVTOR` ima lahko dva atributa – `IME` in `PRIIMEK`:

```
<AVTOR IME="Miha" PRIIMEK="Ficko">
```

HTML predpisuje tako množico uporabljenih označb in pripadajočih atributov kot njihov pomen, deloma pa tudi prikaz podatkov v brskalniku. Na primer `<h1>` je vedno naslov prvega nivoja, označba `<HEADING1>` pa v HTML ni definirana. W3C skupaj s proizvajalci brskalnikov nenehno razširja definicijo HTML, s čimer omogoča vedno večjo raznolikost v načinu predstavitve spletnega dokumenta. Vendar pa so spremembe vedno usklajene z zmožnostmi posameznih brskalnikov.

XML za razliko od HTML ne predpisuje niti označb niti atributov, tudi njihov pomen je lahko poljuben. Označbe so uporabljene le kot ločila med posameznimi deli podatkov. XML je podobno kot SGML le metajezik za opis označevalnih jezikov. Povedano z drugimi besedami, XML ne vsebuje mehanizmov, ki bi predpisovali označbe ali relacije med njimi. Interpretacija označb, s tem pa tudi pomen dokumenta, je v celoti določena z aplikacijo, ki ga uporablja. Označba `<h1>` v XML ne predstavlja nujno naslova, njen pomen je povsem poljuben.

Pravila XML

Če XML ne predpisuje označb, pa je za razliko od HTML doslednejši pri upoštevanju določenih pravil, ki se nanašajo na njihovo sintakso. Najpomembnejše omejitve so:

1. Na najvišjem nivoju je lahko le ena označba, to je korenski element.
2. Imena označb in atributov so občutljiva na velike in male črke. Zato sta `<Avtor>` in `<AVTOR>` različni označbi.
3. Vsaka začetna označba mora imeti pripadajočo končno označbo. Nepravilno:

```
<BODY>
  <P>
    Prvi odstavek ...
  <P>
    Drugi odstavek ...
</BODY>
```

   Označbama `<P>` manjkata končni označbi. Pravilno bi bilo:

```
<BODY>
  <P>
    Prvi odstavek ...
  </P>
  <P>
    Drugi odstavek ...
  </P>
</BODY>
```

4. Označbe morajo biti pravilno gnezdene. Prekrivajoče se označbe niso dovoljene. Nepravilno:

```
<AVTORJI>
  <AVTOR>
    Miha Ficko
  </AVTOR>
  <AVTOR>
    Janez Novak
  </AVTORJI>
</AVTOR>
```

   Končna označba `</AVTORJI>` mora biti pred `</AVTOR>`. Pravilno je torej:

```
<AVTORJI>
  <AVTOR>
    Miha Ficko
  </AVTOR>
```

```
  <AVTOR>
    Janez Novak
  </AVTOR>
</AVTORJI>
```

5. Vrednosti atributov je potrebno vedno navajati v narekovajih. Namesto

   ```
   <AVTOR IME=Miha PRIIMEK=Ficko>
   ```

   je potrebno pisati

   ```
   <AVTOR IME="Miha" PRIIMEK="Ficko">
   ```

Pri HTML lahko v splošnem omenjena pravila kršimo. Tudi če uporabljamo neveljavne označbe, bo brskalnik preprosto preskočil ukaze, ki jih ne razume. V uradni specifikaciji XML pa je celo poudarjeno, da aplikacije, ki uporabljajo XML, ne smejo popravljati napak v datotekah XML. Če dokument ni v skladu s predpisi, se mora pregledovanje ustaviti natanko na tistem mestu, kjer je prišlo do napake.

Vsak dokument XML, ki upošteva vsa zgoraj našteta pravila in ima tako pravo strukturo ga imenujemo dobro formuliran dokument (angl. *Well Formed XML*). Veljaven dokument XML (angl. *Valid XML*) pa imenujemo tisti dokument XML, ki je dobro formuliran in katerega struktura ustreza definiciji DTD (*Document Type Definition*). Dokument DTD vsebuje oznake, ki definirajo relacije med elementi v dokumentu XML. Namesto dokumentov DTD pa se vse bolj uporabljajo Sheme XML.

Tudi pri pisanju spletnih strani se lahko držimo naštetih omejitev in izdelamo dokument HTML v skladu z XML. Tovrstni dokument HTML je tako primer dokumenta XML. XHTML je definiran v specifikaciji konzorcija W3C kot različica standarda HTML 4, ki se drži omejitev XML.

10.5.2 Gradniki dokumenta XML

Element

Elementi so osnovni sestavni deli označevalnega jezika. Element XML je sestavljen iz začetne in končne označbe, ki oklepata določene podatke. Ime elementa navadno opisuje vsebino vsebovanih podatkov. Podatki lahko vsebujejo nove gnezdene elemente:

Primer:

```
<?xml version="1.0"encoding="ISO-8859-2"?>
<seznamUPO>
  <STUDENT>Miha Ficko</STUDENT>
  <STUDENT>Aleš Leban</STUDENT>
  <STUDENT>Monika Kravos</STUDENT>
</seznamUPO>
```

Prva vrstica dokumenta XML je obvezna in določa verzijo in kodni nabor, ki ji priprada preostali del dokumenta. V našem primeru dokument ustreza specifikaciji 1.0 in uporablja kodni nabor ISO Latin 2. Naslednja vrstica: `<seznamUPO>` določa koransko oznako dokumenta. V našem primeru označuje seznam študentov pri predmetu UPO. Preostale vrstice opisujejo podoznake korenske oznake in določajo posamezne študente. Zadnja vrstica: `</seznamUPO>` pa zaključuje korensko oznako.

Atribut

Element lahko vsebuje tudi enega ali več atributov. Atribut je par, sestavljen iz imena in vrednosti, ki sta ločena z zankom "=". Nastopa v začetni označbi takoj za imenom elementa:

Primer:

```
<?xml version="1.0"encoding="ISO-8859-2"?>
<seznamUPO>
  <STUDENT ocena="8">Miha Ficko</STUDENT>
  <STUDENT ocena="9">Aleš Leban</STUDENT>
  <STUDENT ocena="7">Monika Kravos</STUDENT>
</seznamUPO>
```

V zgornjem primeru smo posameznemu študentu dodali atribut, ki označuje njegovo oceno na izpitu. Če bi hoteli dodati še oceno vaj, bi dodali še atribut `ocena_vaj`:

```
<STUDENT ocena="9" ocena_vaj="10">Aleš Leban</STUDENT>.
```

Atribut ali oznaka?

V prejšnem primeru smo študentom priredili ocene, tako da smo dodali ustrezne atribute. Čeprav so v dokumentih HTML atributi oznak zelo priročni pa se jih v dokumentih XML raje izogibamo. Popravimo zgornji primer, da bo vseboval samo oznake:

Primer:

```
<?xml version="1.0"encoding="ISO-8859-2"?>
<seznamUPO>
```

```
  <STUDENT>
    <IME>Miha</IME>
    <PRIIMEK>Ficko</PRIIMEK>
    <OCENA_IZPIT>8</OCENA_IZPIT>
    <OCENA_VAJE>9</OCENA_VAJE>
  </STUDENT>
  <STUDENT>
    <IME>Aleš</IME>
    <PRIIMEK>Leban</PRIIMEK>
    <OCENA_IZPIT>9</OCENA_IZPIT>
    <OCENA_VAJE>10</OCENA_VAJE>
  </STUDENT>
  <STUDENT>
    <IME>Monika</IME>
    <PRIIMEK>Kravos</PRIIMEK>
    <OCENA_IZPIT>7</OCENA_IZPIT>
    <OCENA_VAJE>8</OCENA_VAJE>
  </STUDENT>
</seznamUPO>
```

Uporaba atributov je koristna v primeru, da hočemo neki oznaki dodati informacijo, ki je pomembna samo za program, ki bo podatek uporabil. Na primer: `<SLIKA tip="gif">moja_slika.gif</SLIKA>`

V našem primeru pa bi bilo smiselno dadati posameznemu študentu atribut `ID`, ki bi določal vpisno številko študenta. Med atributi in oznakami obstajajo določene vsebinske razlike, ki jih je potrebno vedeti in upoštevati:

- Atribut se lahko znotraj oznake pojavi samo enkrat, oznaka pa se lahko znotraj druge oznake pojavi večkrat (v našem primeru se oznaka `<STUDENT>` pojavi večkrat).
- Sprememba atributov zahteva večje posege v programsko opremo že delujočih sistemov. Atributi namreč ne strukturirajo podatkov, oznake pa jih.

Referenčna entiteta

Da lahko ločimo gradnike označevalnega jezika od vsebine dokumenta, so določeni znaki rezervirani za označbe. Tako na primer znaka "`<`" in "`>`" določata začetek in konec označbe za element. Če želimo rezervirane znake vključiti v vsebino dokumenta, moramo najti alternativni mehanizem. XML uporablja v ta namen referenčne ali sklicevalne entitete (angl. *entity references*), ki se uporabljajo za predstavitev rezerviranih znakov.

Vsaka entiteta mora imeti enolično določeno ime. Nekatera imena so vnaprej definirana – na primer `lt`, ki določa znak "`<`", razširimo pa jih

lahko tudi s svojimi. Sklic na entiteto se podobno kot v HTML začne z znakom "&" in konča z ";".

Komentar

Komentarje podobno kot v HTML oklepata zaporedji znakov <!-- in -->. Nahajajo se lahko na poljubnem mestu v dokumentu.

Razdelek CDATA

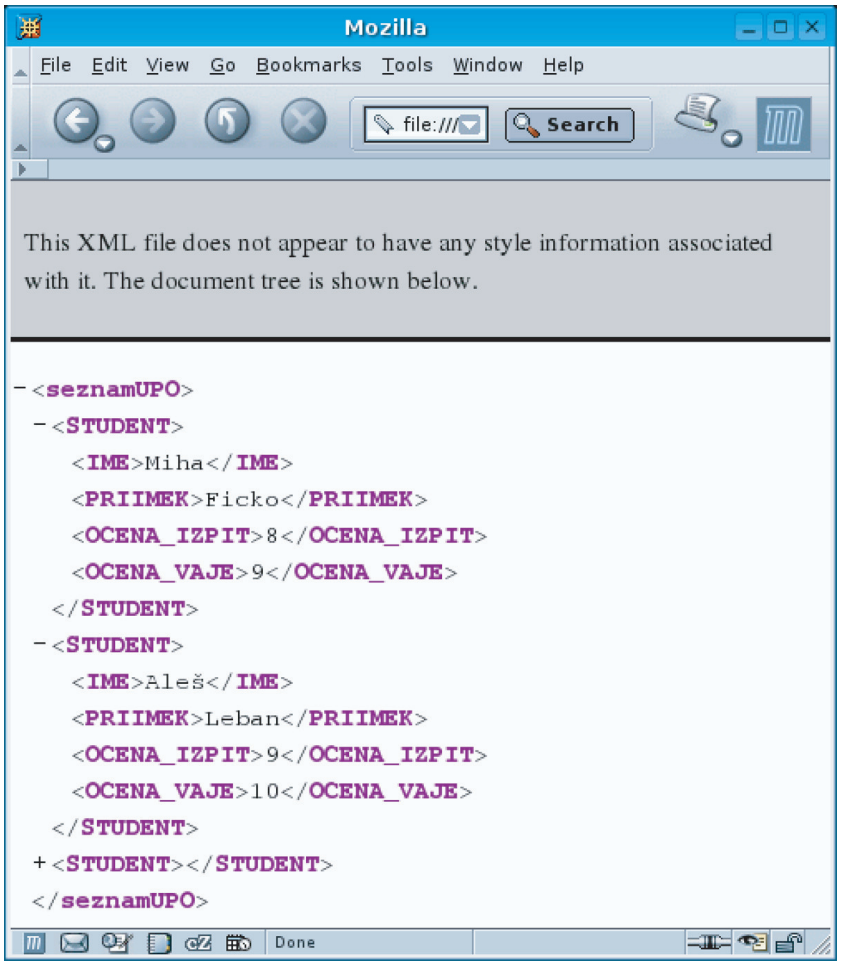
Razdelek CDATA je del dokumenta XML, v katerem razpoznavalnik (angl. *parser*) XML ignorira vse ukaze označevalnega jezika. Vsi znaki, tudi rezervirani, so v tem razdelku obravnavani kot vsebina dokumenta. Razdelek je še posebej uporaben za vključevanje izvorne kode, ki pogosto vsebuje znake kot so na primer "<", ">" in "&":
Primer:

```
<skripta>
<![CDATA[
function matchwo(a,b)
{
if (a < b && a < 0) then
   {
   return 1
   }
else
   {
   return 0
   }
}
]]>
</skripta>
```

Razpoznavalnik XML preskoči vse znake med "<![CDATA[" in "]]>" ter jih neposredno preda aplikaciji brez interpretiranja. Edino znakovno zaporedje, ki se ne sme pojaviti v razdelku CDATA, je "]]>".

10.5.3 Prikaz dokumentov XML

Dokument XML ne nosi nikakršne informacije o tem, kako naj se posamezni podatki prikažejo v brskalniku. Če poskusimo odpreti primer iz prejšnega poglavja v brskalniku, se bo ta prikazal kot navadno besedilo. Nekateri brskalniki pa imajo ugrajen poseben razpoznavalnik, ki prikažejo dokument v drevesni strukturi (slika 10.19).



Slika 10.19: Primer dokumenta XML v brskalniku Mozilla

Če hočemo sami vplivati na videz dokumenta XML na zaslonu moramo uporabiti slogovni dokument XSL (angl. *eXtensible Stylesheet Language*). Jezik XSL sestavljajo trije deli:

- jezik XSLT (angl. *XSL Transformations*), ki služi za preslikavo,
- jezik XPath, ki določa posamezne dele dokumenta XML in
- jezik XSL-FO ali krajše XSL, s katerim določimo končno obliko dokumenta XML.

Na jezik XSL lahko tako gledamo kot na jezik, s pomočjo katerega preslikamo dokument XML v drug dokument XML, pri tem lahko posamezne dele dokumeta preurejamo, dodajamo ali spremenimo. Prikaz lahko določimo tudi na podlagi vrednosti podatkov v samem dokumetu. Prikaz končnega dokumeta lahko priredimo za različne medije kot so

papir, zaslon ali kot zvok. V nadaljevanju bomo spoznali najvažnejši del jezika XSL, jezik XSLT, ki služi za preslikavo dokumenta XML v drug dokument XML ali, v našem primeru, v dokument HTML, da ga bomo lahko prikazali v spletnem brskalniku.

Primer dokumeta XSL:

```
<?xml version="1.0" encoding="ISO-8859-2"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <html>
  <body>
    <h2>Rezultati pisnega izpita UPO</h2>
    <table border="1">
    <tr bgcolor="#9acd32">
      <th align="left">Ime</th>
      <th align="left">Priimek</th>
      <th align="left">Ocena izpita</th>
    </tr>
    <xsl:for-each select="seznamUPO/STUDENT">
    <tr>
      <td><xsl:value-of select="IME"/></td>
      <td><xsl:value-of select="PRIIMEK"/></td>
      <td><xsl:value-of select="OCENA_IZPIT"/></td>
    </tr>
    </xsl:for-each>
    </table>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

Korenska oznaka: `<xsl:stylesheet>` označuje, da gre za dokument XSL. Atribut: `xmlns:xsl="http://www.w3.org/1999/XSL/Transform"` označuje njen imenski prostor, kot ga določa konzorcij W3C. Pri uporabi tega imenskega prostora moramo obvezno navesti tudi atribut `version="1.0"`, ki določa različico.

Oznaka `<xsl:template>` določa začetek vzorca. Vsak dokument XSL je namreč sestavljen iz različnih pravil, ki jih imenujemo vzorci. Atribut `match="/"` določi, da naš vzorec zavzema celoten dokument XML. Preostali del dokumenta pripada temu vzorcu, razen zadnjih dveh vrstic, ki zaključujeta vzorec in sam dokument XSL.

V dokument dodamo HTML oznake, ki določajo naslov in začetek tabele.

Vsebino tabele pa določimo s pomočjo oznak jezika XSL. Naslednja pomembena oznaka je `<xsl:for-each>`, s pomočjo katere izberemo vse elemente dokumenta XML, ki jih določa atribut `select`. V našem primeru izberemo vse elemente pod oznakama `"seznamUPO/STUDENT"`. Sintaksa, ki jo uporablja atribut `select`, je določena z jezikom XPath, ki je sestavni del jezika XSL. Pri izbiri elementov lahko tako uporabljamo vsa pravila, ki jih določa jezik XPath. Na primer, če bi hoteli izbrati samo študente, ki imajo oceno izpita večjo od 8, bi zapisali:
`<xsl:for-each select="seznamUPO/STUDENT[OCENA_IZPIT>8]">`

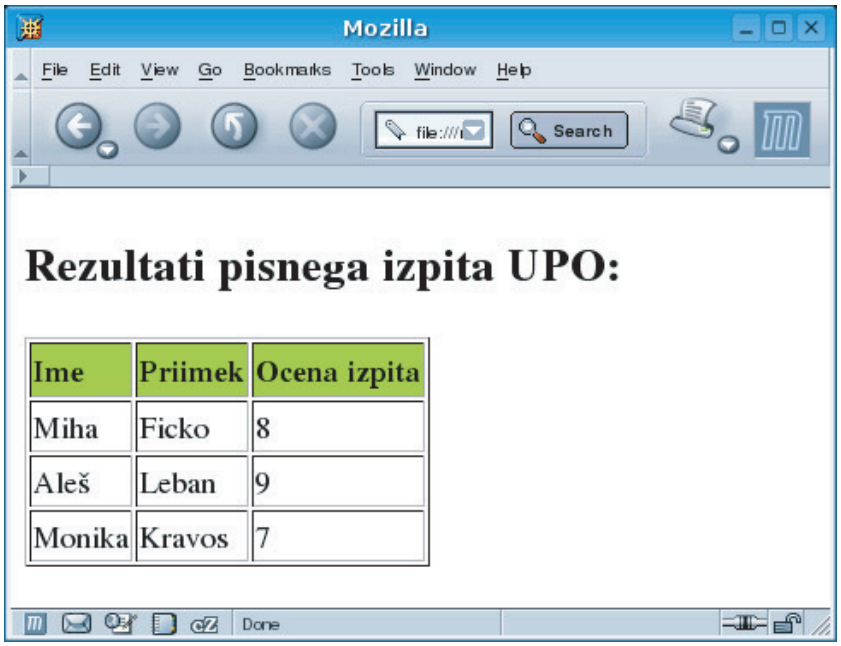
S pomočjo ukaza `<xsl:value-of>` pa izberemo določen element iz dokumenta XML, ki ga želimo prikazati na izhodu. V našem primeru želimo prikazati vrednosti oznak `IME, PRIIMEK` in `OCENA_IZPIT`. Tukaj lahko tudi opazimo lepo lastnost jezika XML, to je razširljivost. Čeprav smo XML dokument kasneje razširili še z oznako `OCENA_VAJE`, to nič ne vpliva na delovanje našega programa oziroma na videz dokumenta XML.

Če pa hočemo, da zgoraj določena pravila dokumenta XSL vplivajo na videz našega dokumenta XML, moramo v dokumentu XML dodati še vrstico, ki označuje kje se nahaja datoteka XSL. Če zgornji primer dokumenta XSL shranimo v datoteko z imenom `ocena.xsl`, moramo v dokument XML dodati sledečo vrstico:
`<?xml-stylesheet type="text/xsl" href="ocena.xsl"?>`. (Včasih ima brskalnik težave s prepoznavo datotek s končnico xsl. V tem primeru preimenujmo datoteko `ocena.xsl` v `ocena.xml` in ustrezno popravimo vrstico v izvornem dokumentu XML.)

Slika 10.20 prikazuje primer dokumenta XML, ki smo mu določili obliko s pomočjo dokumenta XSL.

Seveda pa lahko s pomočjo jezika XSL naredimo veliko več. V naslednjem primeru bomo seznam študentov razvrstili po priimkih in posebej označili tiste študente, ki so dosegli oceno izpita višjo od 7 in oceno vaj višjo od 9. Za razvrščanje bomo uporabili ukaz XSL: `<xsl:sort/>` in mu z atributom `select` določili, po kateri oznaki bomo seznam uredili. Za izpis boljših študentov v drugačni obliki pa bomo uporabili ukaz: `<xsl:choose>`, ki v kombinaciji z ukazoma: `<xsl:when>` in `<xsl:otherwise>` omogoča različne odločitve glede na predhodno izpolnjeni pogoj, ki ga navedemo v atributu `test`. Poglejmo si omenjene ukaze na primeru:

```
<?xml version="1.0" encoding="ISO-8859-2"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <html>
  <body>
```



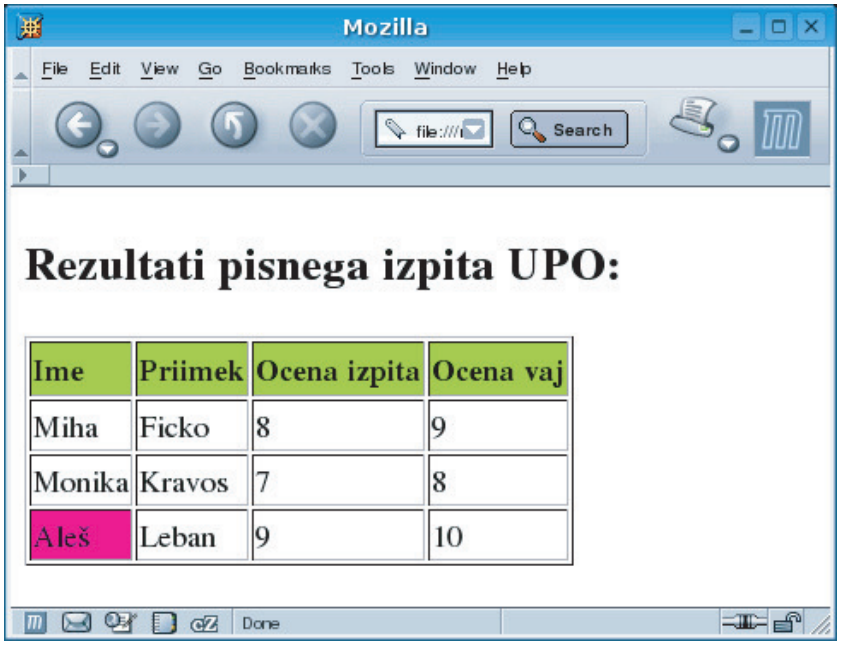
Slika 10.20: Videz dokumenta XML, ki smo mu določili poseben videz

```
<h2>Rezultati pisnega izpita UPO:</h2>
<table border="1">
<tr bgcolor="#9acd32">
  <th align="left">Ime</th>
  <th align="left">Priimek</th>
  <th align="left">Ocena izpita</th>
  <th align="left">Ocena vaj</th>
</tr>
<xsl:for-each  select="seznamUPO/STUDENT">
<xsl:sort select="PRIIMEK"/>
<tr>
  <xsl:choose>
    <xsl:when test="OCENA_IZPIT>7  and OCENA_VAJE>9">
      <td bgcolor="#ff00ff"><xsl:value-of select="IME"/></td>
    </xsl:when>
    <xsl:otherwise>
      <td><xsl:value-of select="IME"/></td>
    </xsl:otherwise>
  </xsl:choose>
  <td><xsl:value-of select="PRIIMEK"/></td>
  <td><xsl:value-of select="OCENA_IZPIT"/></td>
  <td><xsl:value-of select="OCENA_VAJE"/></td>
</tr>
</xsl:for-each>
```

```
    </table>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

S pomočjo ukaza za urejanje smo uredili seznam po priimkih. Atribut `test` oznake `<xsl:when>` pa smo določili na vrednost: `"OCENA_IZPIT > 7 and OCENA_VAJE > 9"`.

Prikaz dokumenta XML, ki uporablja tako definirano slogovno datoteko XSL, prikazuje slika 10.21.



Slika 10.21: Urejeni seznam z označenim najboljšim študentom

10.5.4 Sklop tehnologij okoli XML

V prejšnem poglavju smo spoznali dokumente XSL, ki določajo videz samih dokumentov XML. Okoli XML pa je zrasel velik sklop še drugih tehnologij, ki so s specifikacijo standarda najtesneje povezane. Z opredelitvijo pomena posebnih označb in atributov omogočajo različne naloge.

Shema XML (angl. *XML Schema*) predpisuje zgradbo določenega dokumenta XML. S predpisom omejimo nabor označb in atributov, ki lahko nastopajo v dokumentu, določimo pa tudi njihovo hierarhično vsebovanost. S shemo XML lahko tako na primer predpišemo obliko elektronskega računa v obliki XML, ki se uporablja v podjetju.

XSL (angl. *eXtensible Stylesheet Language*) oziroma njegov glavni del XSLT (angl. *XSL Transformations*) je jezik na osnovi XML, ki omogoča preslikavo med različnimi dokumenti XML. S preslikavo lahko preuredimo, dodamo ali spremenimo posamezne označbe in atribute v dokumentu XML. S tem lahko na primer omogočimo izmenjavo vsebinsko enakovrednih elektronskih računov med posameznimi podjetji, tudi če uporabljajo različne oblike dokumentov. Ker je HTML primer XML, lahko z XSLT določimo tudi preslikavo v HTML. Na ta način lahko podatkom v dokumentu XML določimo preslikavo v spletni dokument, s čimer se XML drži načela ločevanja vsebine od oblike.

XLink opisuje standardni način za dodajanje hipertekstnih povezav v datoteko XML.

XPointer in **XFragments** predpisujeta sintakso za sklicevanje na posamezne dele dokumenta XML. Za razliko od URL se lahko sklicujemo na točno določene podatke v dokumentu XML.

XML **DOM** (angl. *Document Object Model*, dokumentni objekni model) je standardni nabor programskih vmesnikov za delo z dokumenti XML iz različnih programskih jezikov.

Imenski prostor XML (angl. *XML Namespace*) predpisuje, kako z URL določiti imenski prostor za posamezno označbo ali atribut v dokumentu.

XML predstavlja z vsemi okoliškimi tehnologijami precej zajetno snov, ki je zaradi posebne terminologije, ki se uporablja, v začetku precej zahtevna za branje. Na tem mestu se ne bomo spuščali v podrobnosti nobene od tehnologij. Velja pa opozoriti, da je za učinkovito reševanje problemov na osnovi XML zaradi njihove tesne medsebojne povezanosti potrebno poznati vsaj osnovne mehanizme vseh tehnologij okoli XML. Videli pa smo, da lahko zajeten kos problemov rešimo že z grobim poznavanjem osnovne specifikacije XML in XSLT. Dobro je poznati še sheme XML in če želimo z dokumenti XML delati še v programskih jezikih, pa si moramo ogledati tudi ustrezne vmesnike XML DOM.

XML s pripadajočimi moduli je podobno kot HTML v nenehnem razvoju. Funkcionalnost že standardiziranih modulov se neprestano razširja, pojavljajo pa se tudi novi. Zaradi tega je priporočljivo redno spremljati dogajanje na spletnem mestu konzorcija W3C:
`www.w3.org/XML`.

10.6 Objava spletnih strani na spletnem strežniku

Če hočemo, da bodo naše spletne strani videli še drugi, ki so priključeni v svetovno omrežje internet, jih moramo prenesti iz lokalnega računalnika

na spletni strežnik.

Spletni strežnik je računalnik, ki je vedno priključen na internet in na katerem poseben program gosti spletne strani, ki jih na zahtevo pošlje uporabniku. Seveda je to samo osnovna naloga spletnega strežnika. Poskrbeti mora še za izvajanje skriptnih programov, za omejevanje dostopa, za varen prenos itd.

Spletne strani običajno prenesemo na spletni strežnik s pomočjo programa za prenos datotek FTP (*File Transfer Protocol*). Najpogostejši spletni strežnik je Apache, ki privzeto zahteva, da ima glavna stran ime `index.html`. Če se bodo naše spletne strani nahajale na operacijskem sistemu Unix oziroma Linux, kar je za spletni strežnik Apache običajno, moramo tudi paziti, da imamo vsa imena datotek HTML napisana tako, kot se na njih sklicujemo v spletnem dokumentu. Sistem Linux namreč loči med velikimi in malimi črkami, kar pri sistemih Windows ni pomembno. Po prenosu moramo tudi preveriti, da imajo vsi uporabniki pravico do branja datotek HTML. Običajno se datoteke nahajajo v mapi uporabnika in sicer v podmapi `public_html` ali `www` (na primer `/home/mihanovak/public_html`). Če nimamo svojega spletnega strežnika, obstaja nekaj brezplačnih, ki bodo v zameno na naših straneh prikazovali oglase. Eden takšnih strežnikov je `www.geocities.com`.

10.7 Sistemi za upravljanje z vsebinami

Pri obvladovanju večjih spletnih mest si lahko pomagamo s *sistemi za upravljanje vsebin*, navadno jih označujemo s kratico CMS (*Content Management System*). Navadno so to programi, nameščeni na spletnem strežniku, ki omogočajo bolj ali manj udobno urejanje spletnega mesta kar preko spletnega brskalnika. Sistemov za upravljanje vsebin je mnogo in izbira je včasih težavna.

Posebej zanimiv sistem za upravljanje vsebin je *Wiki*. Beseda prihaja s Havajev in pomeni *hitro*. Wiki je sestavljen iz *člankov* – vsak članek ustreza spletni strani – do katerih dostopamo s pomočjo *gesel* (angl. *Wikiwords*). Vsako geslo je unikatno za celoten Wiki in nedvoumno določa članek. Gesla uporabljamo namesto naslovov URL za tvorjenje povezav na članek. Sistem je organiziran podobno kot enciklopedije in slovarji in zato še posebej primeren za tovrstne vsebine.

Na Wikiju ima vsaka stran gumb "uredi", s katerim uporabnik prikliče urejevalnik. Vsebina strani je napisana v preprostem označevalnem jeziku: `[[geslo]]` ustvari povezavo na drugo stran ali vložek (npr. sliko), `''poudarjeno''` piše poudarjeno, `'''krepko'''` piše krepko, itd... Sintaksa močno spominja na oznake, ki jih uporabniki radi uporabljajo v elektronski pošti. Točna sintaksa je odvisna od uporabljenega Wikija in je

na voljo kot pomoč v urejevalniku.

Wiki je hiter zato, ker omogoči:

1. Takojšnje urejanje. Urejevalnik je dostopen z enim klikom.
2. Uporabniku ni potrebno razmišljati o imenikih, naslovih URL in datotekah, temveč o geslih in člankih, ki so višji koncepti.
3. Enostavno sodelovanje več avtorjev.
4. Dnevnik sprememb za vsak članek. Možno je priklicati katerokoli prejšnjo verzijo članka.
5. Enciklopedično naslavljanje. Uporabniku se ni potrebno posebej ukvarjati s strukturo spletnega mesta, izbirati mora le primerna ustrezna gesla, kar je lažje.
6. Enostaven označevalni jezik, ki je za tipkanje učinkovitejši kot HTML.
7. Celovito oblikovanje videza celotnega spletnega mesta s pomočno predloge.

Na spletu je precej mest Wiki, eden najpopularnejših je *Wikipedija*. Wikipedija je brezplačna enciklopedija in vsebuje vrsto znanja iz najrazličnejših področij v več kot 300.000 člankih. Poleg angleške verzije, ki je največja, obstaja tudi v skoraj vseh svetovnih jezikih – tudi v slovenščini. Na vsakem članku so povezave na ustrezne članke v drugih jezikih – tako da jo lahko uporabljamo tudi kot nekakšen slovar. Wikipedijo poganja sistem MediaWiki.

10.8 Registracija svoje domene

Ko postavimo svoje spletne strani na strežnik, dostopamo do njih običajno kot: `http://www.streznik.domena.net/~janezn/`. Če hočemo do svojih spletnih strani dostopati preko enostavnejšega naslova, moramo določiti svojo domeno (na primer `www.novak.net`). Najprej moramo preveriti ali je domena s takim imenom prosta. To preverimo v tako imenovani bazi WHOIS, kjer tudi dobimo podatke, kdo je lastnik domene v primeru, če je domena zasedena. Nekatere domene lahko pridobimo tudi brezplačno (na primer domene s končnico `.tk`), nekatera podjetja ponujajo brezplačno določitev domene za poskusno obdobje, večinoma pa moramo svojo domeno plačati za obdobje najmanj enega leta. Znesek plačila za zakup same domene ni visok, se pa poveča, če hočemo poleg registracije svoje domene še katero drugo storitev, kot je gostovanje spletnih strani, registracija dodatnih poštnih naslovov, itd.

Komercialne domene, ki jih ponujajo tako domači kot tuji ponudniki, so običajno s končnicami `.com`, `.net`, `.org`, `.biz`, `.info`, `.tv` itd. Domene, ki pripadajo posameznim državam (na primer `.de`, `.ru`, `.it`, `.si`), pa niso javno dostopne in jih lahko pridobijo samo uporabniki pod posebnimi pogoji države. Tako za slovensko domeno `.si` skrbi javni zavod ARNES. Slovensko domeno s končnico `.si` si lahko pridobijo samo pravne osebe in sicer z naslovom registrirane organizacije.

Pri registraciji domene običajno lahko določimo preusmeritev na spletni strežnik kjer gostimo svoje spletne strani ali pa številki IP domenskega strežnika. Za popolni nadzor nad domeno je priporočljivo, da imamo postavljena imenska strežnika DNS, ki bosta določala spletni strežnik. En strežnik je primarni, drugi pa sekundarni. Na ta način lahko svoji domeni določimo tudi poddomene in poljubne poštne naslove. Tako lahko pod registrirano domeno `novak.net` dodamo še poddomene `www`, `janez`, `metka`, ki bodo določali naslove: `www.novak.net`, `janez.novak.net` in `metka.novak.net`. Dodamo si lahko tudi poljubne poštne naslove pod svojo domeno, kot na primer: `janez@novak.net` ali `metka@novak.net`.

10.9 Naloge

1. Katere tri grafične zapise najpogosteje uporabljamo za vključitev slik na spletno stran? Za kakšne vrste slik se uporabljajo?
2. Katere štiri osnovne strukture spletnih strani poznamo in na podlagi česa jih izberemo?
3. Kaj lahko razberete iz naslova URL: `http://cogvis.fri.uni-lj.si/pub/seznam.html`?
4. Kako najlažje dobimo informacijo v svetovnem spletu? Ali lahko izberemo tudi format strani, po katerih iščemo, in kako?
5. Naslova katerih dveh strežnikov moramo poznati pri določevanju elektronske pošte in za kaj se kateri od njiju uporablja?
6. V jeziku HTML izdelaj tabelo s seznamom študentov, ki so opravili izpit. Tabela naj ima naslov in vsebuje različno označene študente, ki so dosegli oceno višjo od 9.
7. Napiši preprost dokument XML, ki bo vseboval dve pismi `<PISMO>`. Pismo naj vsebuje prejemnika, pošiljatelja in naslov.
8. Izdelaj dokument XSL, ki bo dokument XML iz prejšnje naloge pretvoril v tabelo dokumenta HTML.
9. Izdelaj dve spletni strani in glavno stran `index.html`, ki bo obe strani prikazala v okvirjih.

10. Na spletno stran vključi sliko, ki bo vsebovala povezavo na nov dokument HTML.

10.10 Koristne spletne povezave

1. Šola internetnih tehnologij: HTML, XHTML, CSS, XML, XSL itd: `http://www.w3schools.com`
2. Domača stran W3C za HTML: `http://www.w3.org/MarkUp/`
3. Kako deluje spletni strežnik? `http://www.howstuffworks.com/web-server.htm`
4. Kako deluje elektronska pošta? `http://www.howstuffworks.com/email.htm`
5. Domača stran W3C za XML: `http://www.w3.org/XML`
6. Domača stran založbe O'Reilly: `http://www.xml.com`
7. Tečaji internetnih tehnologij; bazirani na primerih: `http://www.zvon.org`
8. Apache spletni strežnik: `http://www.apache.org`
9. Wikipedia, prosta enciklopedija: `http://www.wikipedia.org/`
10. Wikipedija, slovenska prosta enciklopedija: `http://sl.wikipedia.org/`

10.11 Literatura

[1] E. Barrett, editor. *Text, ConText, and HyperText.* MIT Press, Cambridge, MA, 1988.

[2] J. Nielsen. *Designing Web Usability.* New Riders, Indianapolis, IN, 2000.

[3] D. Siegel. *Secrets of Successful Web Sites.* Hayden Books, Indianapolis, 1997.

[4] D. Siegel. *Creating Killer Web Sites.* Hayden Books, Indianapolis, second edition, 1998.

[5] E. R. Tufte. *The Visual Display of Quantitative Information.* Graphics Press, Cheshire, CT, 1983.

[6] E. R. Tufte. *Envisioning Information.* Graphics Press, Cheshire, CT, 1990.

[7] E. R. Tufte. *Visual Explanations.* Graphics Press, Cheshire, CT, 1997.

[8] J. Veen. *The Art & Science of Web Design.* New Riders, Indianapolis, IN, 2000.

[9] M. J. Young. *Step by Step XML.* Microsoft Press, Redmond, WA, 2000.

11

Primeri uporabe

V tem poglavju bomo pokazali, kako lahko različna orodja med seboj povezujemo pri izračunih, pisanju poročila in izdelavi govorne predstavitve. Rdeča nit bo priprava poročila in predstavitve seminarske naloge na temo superkvadrikov. Za pisanje poročila bomo uporabili LaTeX, v katerega bomo vključili obrazce in slike v formatu PostScript, ki bodo prikazovale podatke, pridobljene v programu Octave in obdelane v programu OpenOffice.org Calc. Predstavitev bomo izdelali v programu OpenOffice.org Impress.

Vsa potrebna orodja najdemo na priloženi zgoščenki kot del namestitve operacijskega sistema Slix. Vsa orodja so praviloma tudi del najpogostejših distribucij Linuxa. Izjema je morda le Octave, ki se v operacijskem sistemu Linux pogosto uporablja kot nadomestek za Matlab, saj ima praktično identično sintakso, v nasprotju z Matlabom pa spada med brezplačno odprto programje. Če naša distribucija Linuxa ne vsebuje paketa Octave, si ga lahko prenesemo s svetovnega spleta.

11.1 Opis naloge

S superkvadriki smo se srečali že v poglavju o matematičnih programih, kjer smo pri opisu programskega paketa Matlab napisali funkcijo za izris superkvadrika. Superkvadrike ponavadi uporabljamo za parametrične opise tridimenzionalnih oblik v računalniškem vidu in v računalniški grafiki. Z laserskim ali svetlobnim globinskim senzorjem lahko naprimer izmerimo oddaljenost točk na površini nekega predmeta in nato izračunamo parametre superkvadrika, ki najbolje opiše oblak točk v tridimenzionalnem prostoru [3].

Denimo, da imamo podan oblak točk in dva hipotetična parametrična opisa. Zanima nas, koliko izmerjeni podatki odstopajo od izračunanih geometrijskih modelov. Najprej bomo opisali mero za oddaljenost točke

od površine superkvadrika. Nato bomo za vsako točko izračunali, kakšno je njeno odstopanje do vsakega od dveh modelov. Odstopanja bomo nato statistično analizirali in tako ocenili ustreznost posamičnega superkvadrika.

11.2 Risanje superkvadrikov

Za risanje superkvadrika lahko uporabimo funkcijo `sq`, opisano na strani 296. Tam je opisana tudi eksplicitna enačba, ki opisuje točke na površini superkvadrika:

$$\begin{aligned} x &= a_1 \cos^{\epsilon_1} \eta \cos^{\epsilon_2} \omega, \\ y &= a_2 \cos^{\epsilon_1} \eta \sin^{\epsilon_2} \omega, \\ z &= a_3 \sin^{\epsilon_1} \eta. \end{aligned} \tag{11.1}$$

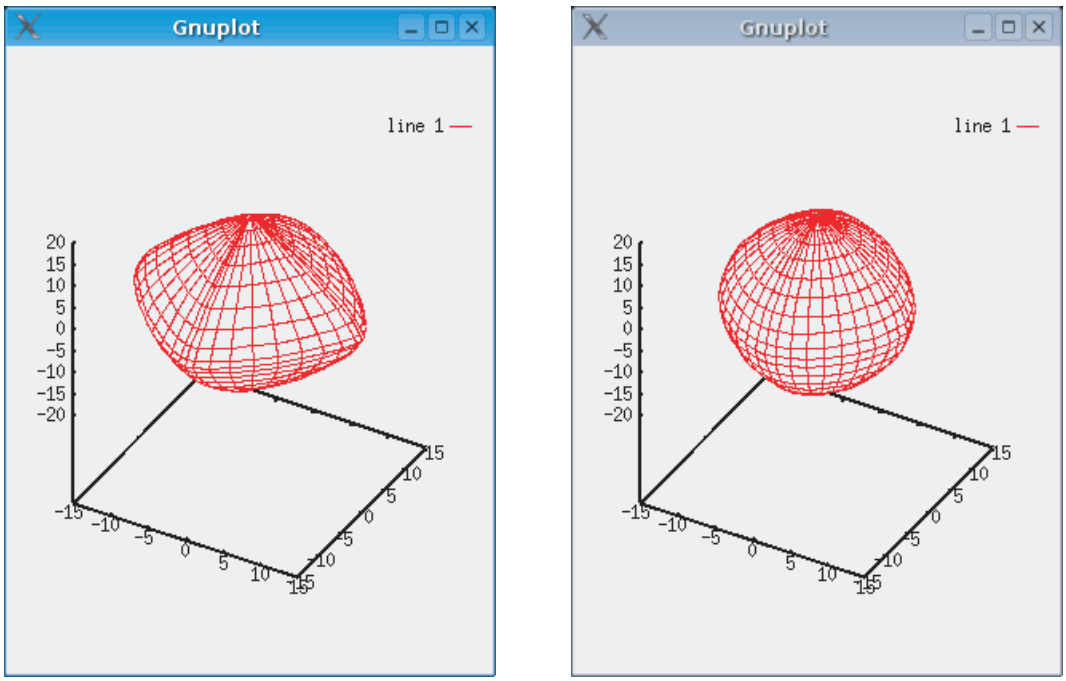
Parametri a_1, a_2 in a_3 določajo velikost superkvadrika, ϵ_1 in ϵ_2 obliko. Enačbe veljajo za superkvadrik v koordinatnem sistemu, katerega središče sovpada s središčem superkvadrika. Opis superkvadrika v splošnem položaju vsebuje še parametre rotacije in translacije, ki pa jih v našem primeru ne bomo obravnavali. Volumetrično telo tako lahko opišemo s samo petimi parametri.

Denimo, da smo kot kandidata za opis oblaka točk izračunali dva superkvadrika. Parametri prvega naj bodo $a_1^1 = 15$, $a_2^1 = 15$, $a_3^1 = 20$, $\epsilon_1^1 = 1.8$ in $\epsilon_2^1 = 1.5$, parametri drugega pa $a_1^2 = 12$, $a_2^2 = 12$, $a_3^2 = 20$, $\epsilon_1^2 = 1.5$ in $\epsilon_2^2 = 1.2$. Obe telesi sta v izhodiščnem položaju; zato moramo funkciji `sq` (str. 296) kot središče podati vektor `C=[0 0 0]`, kot komponente rotacij okrog osi koordinatnega sistema pa `rotX=0, rotY=0` in `rotZ=0`. Ker pa so to že privzete vrednosti za zadnje štiri parametre, jih pri klicu funkcije lahko izpustimo. V ukazno vrstico programa Octave torej vtipkamo:

```
>> sq(15, 15, 20, 1.8, 1.5);
```

Ukaz odpre okno, ki vsebuje grafiko prvega superkvadrika (slika 11.1 levo). Ker bomo grafiko potrebovali za končno poročilo, jo moramo shraniti v datoteko. Za matematične grafe je pomembno, da jih shranjujemo v vektorskem grafičnem zapisu, med drugim tudi zato, da pri spreminjanju velikosti ne bodo izgubili kvalitete.

Preden shranimo grafiko v datoteko moramo vedeti, kako Octave grafike izrisuje. Za to uporablja poseben paket `gnuplot`, ki je del večine distribucij operacijskega sistema Linux. Octave vsebuje nizkonivojski



Slika 11.1: Izris superkvadrikov v okna X Window System

funkciji `gplot` in `gsplot`, ki predstavljata `gnuplot`-ovi funkciji `plot` in `splot`. Vse ostale visokonivojske grafične funkcije so osnovane na klicih tega bazičnega nabora.

Parametre izrisovanja grafik nadzorujemo tako, da spreminjamo parametre programa `gnuplot`. Pri tem uporabljamo funkcijo `gset`, npr.

```
>> gset title "Superkvadrik";
```

izrisani grafiki doda naslov. Po spremembi parametra moramo grafični izris osvežiti z ukazom `replot`.

Privzete nastavitve zagotavljajo, da se bodo grafike izrisovale na unixov grafični terminal, ki je v večini primerov v domeni grafičnega sistema X Window System. Vendar pa lahko Octave grafike izrisuje na katerikoli izhod, ki ga podpira `gnuplot`. Seznam podprtih terminalov dobimo z ukazom

```
>> gset term;
```

Če želimo grafiko shraniti v vektorskem grafičnem formatu PostScript, vtipkamo naslednje:

```
>> gset term postscript;
>> gset output "superkvadrik.ps"; replot;
```

Vektorska slika grafike se sedaj nahaja v datoteki `superkvadrik.ps`. Če imamo tiskalnik, ki podpira format PostScript, lahko grafike preusmerimo neposredno v tiskanje:

```
>> gset term postscript;
>> gset output "|lpr -Pime_tiskalnika"; replot;
```

Ukaz

```
>> gset term x11;
```

preusmeri grafični izhod nazaj v okna X Window System.

11.3 Računanje razdalj točk do superkvadrika

Denimo, da imamo v datoteki `tocke.mat`[1] shranjen oblak točk, ki smo ga opisali z obema superkvadrikoma. Koordinate točk naložimo v Octave z ukazom

```
>> load -ascii tocke.mat;
```

Točke bomo izrisali v tridimenzionalnem prostoru. Ker so točke predstavljene v matriki, moramo ponastaviti parametrični način risanja:

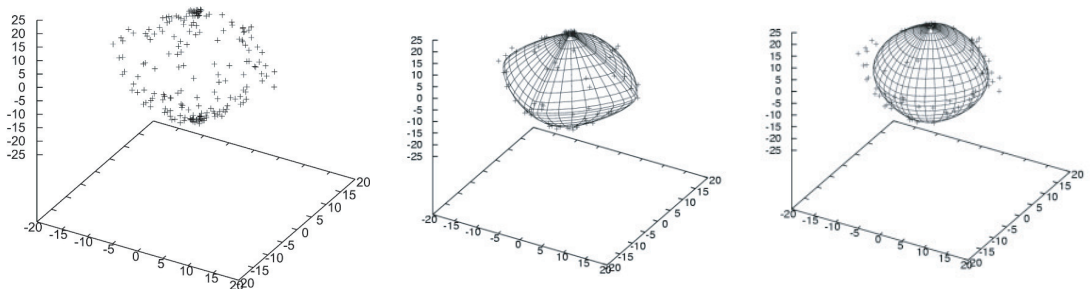
```
>> gset parametric;
>> gsplot(tocke) with points;
>> axis("equal");
>> gset term postscript;
>> gset output "tocke.ps"; replot;
```

Ukaz `axis("equal")` določi lastnosti osi. Rezultat vidimo na sliki 11.2 (a). Sedaj bomo skupaj z oblakom točk izrisali še oba superkvadrika. Uporabili bomo ukaz `hold`, ki prej izrisano grafiko zadrži in preko nje izriše novo.

```
>> sq(15, 15, 20, 1.8, 1.5);
>> axis("equal");
>> hold on;
>> gset parametric;
>> gsplot(tocke) with points;
```

[1] Datoteko `tocke.mat lahko prenesete z naslova http://lrv.fri.uni-lj.si/studij/upo/tocke.txt`

```
>> gset term postscript;
>> gset output "superkvadrik1.ps"; replot;
>> hold off;
>> sq(12, 12, 20, 1.5, 1.2);
>> axis("equal");
>> hold on;
>> gset parametric;
>> gsplot(tocke) with points;
>> gset output "superkvadrik2.ps"; replot;
```



tocke.ps superkvadrik1.ps superkvadrik2.ps

Slika 11.2: Vektorske grafike

Na slikah 11.2 (b) in (c) že lahko vidimo, kako se superkvadrika prilegata oblaku točk. Sedaj moramo poiskati ustrezno mero za oceno primernosti posameznega superkvadrika. Mera je navadno osnovana na statistiki razdalj posameznih točk do površine superkvadrika. Vendar pa je evklidsko razdaljo točke do površine superkvadričnega telesa mogoče izračunati le z numeričnimi metodami, algebraično pa je neizračunljiva [2]. Kot približek pa lahko uporabimo t.i. radialno evklidsko razdaljo, ki je definirana kot razdalja med točko in površino superkvadrika v smeri premice, ki gre skozi točko in skozi center superkvadrika.

Za izračun radialne razdalje potrebujemo enačbo, ki jo najdemo v [2] na strani 21:

$$F(x,y,z) = \left(\left(\frac{x}{a_1} \right)^{\frac{2}{\epsilon_2}} + \left(\frac{y}{a_2} \right)^{\frac{2}{\epsilon_2}} \right)^{\frac{\epsilon_2}{\epsilon_1}} + \left(\frac{z}{a_3} \right)^{\frac{2}{\epsilon_1}} \tag{11.2}$$

Točka (x, y, z) leži na površini superkvadrika, ko je vrednost funkcije F enaka 1. Implicitna enačba superkvadrika je torej

$$\left(\left(\frac{x}{a_1} \right)^{\frac{2}{\epsilon_2}} + \left(\frac{y}{a_2} \right)^{\frac{2}{\epsilon_2}} \right)^{\frac{\epsilon_2}{\epsilon_1}} + \left(\frac{z}{a_3} \right)^{\frac{2}{\epsilon_1}} = 1 \tag{11.3}$$

Ko je F negativen, leži točka v notranjosti superkvadrika. Ko pa je vrednost F pozitivna, leži točka izven superkvadrika. Radialna razdalja točke T s koordinatami $\mathbf{r_0} = (x_0, y_0, z_0)$ do površine superkvadrika pa je definirana kot

$$d = |\mathbf{r_0}||1 - F^{-\frac{\epsilon_1}{2}}(x_0, y_0, z_0)| \ . \tag{11.4}$$

Funkcijo za izračun razdalje d vnesemo v Octave:

```
function d = dsq(x0, y0, z0, a1, a2, a3 ,e1 ,e2)
% function dsq(x0, y0, z0, a1, a2, a3, e1, e2)
% izracuna radialno razdaljo tocke T=(x0, y0, z0) do
% superkvadrika s parametri a1,a2,a3 (velikost superkvadrika),
% e1,e2 (oblika).

F=((x0./a1).^(2/e2)+(y0./a2).^(2/e2)).^(e2/e1)+(z0./a3).^(2/e1);
d=sqrt(x0.^2+y0.^2+z0.^2).*abs(1.-F.^(-e1/2));
```

Oddaljenost točk do prvega in drugega superkvadrika izračunamo z

```
>> d1 = dsq(tocke(:,1), tocke(:,2), tocke(:,3), 15, 15, 20, 1.8, 1.5);
>> d2 = dsq(tocke(:,1), tocke(:,2), tocke(:,3), 12, 12, 20, 1.5, 1.2);
```

Rezultata združimo v enotno matriko `D = [d1 d2]`. Matriko `D` nato shranimo v datoteko `razdalje.txt`:

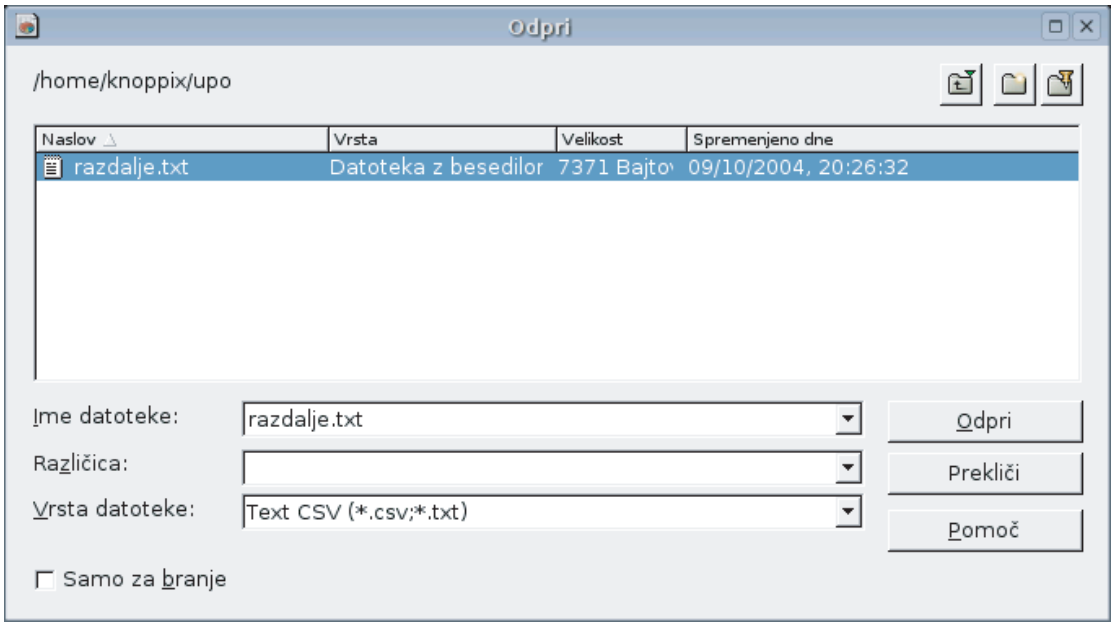
```
>> save -ascii razdalje.txt D
```

11.4 Obdelava podatkov v OpenOffice.org Calc

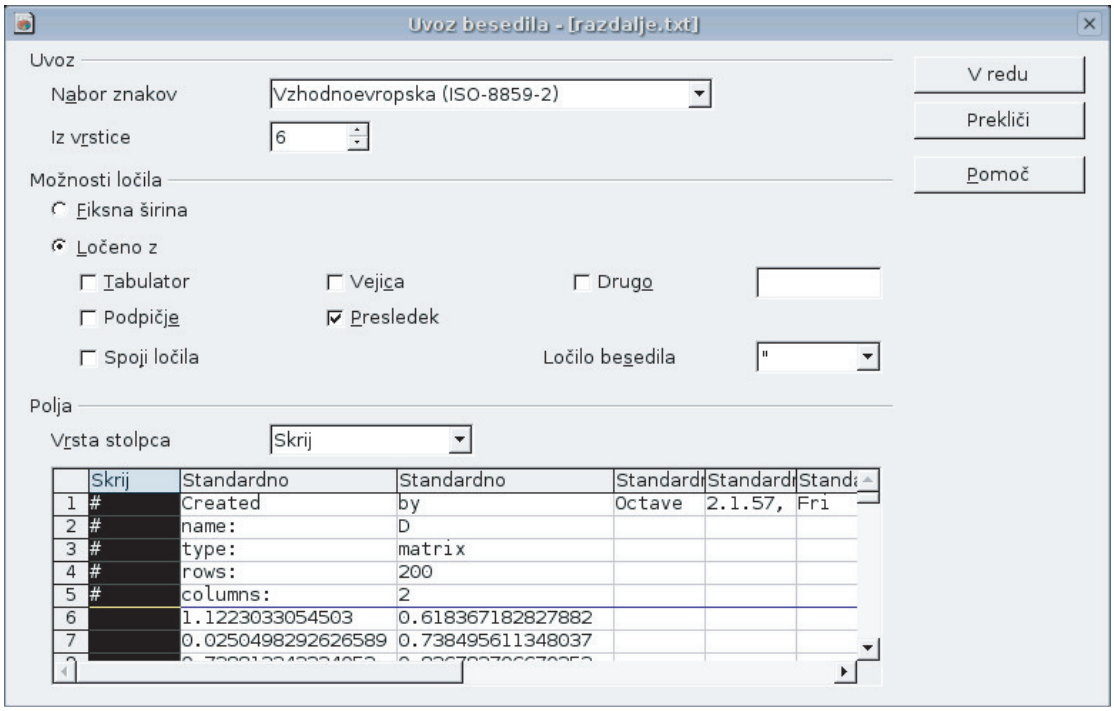
Izračunane razdalje točk do površine obeh superkvadrikov bomo sedaj obdelali in prikazali v programu OpenOffice.org Calc.[2] Po zagonu v meniju "Datoteka" izberemo opcijo "Odpri". V pogovornem oknu kot "Vrsta datoteke" izberemo `text CSV` in poiščemo datoteko `razdalje.txt` (slika 11.3).

V oknu "Uvoz besedila" (slika 11.4) nato vnesemo oznako vrstice, kjer naj se uvoz začne (v našem primeru 6, saj prvih pet vrstic predstavlja metapodatke, ki jih je Octave zapisal ob shranjevanju v zapisu ASCII), ter določimo ločilo, ki označuje prehode med stolpci. Prvi stolpec lahko že pri uvozu skrijemo. V tako dobljeno tabelo vrinemo prvo vrstico, v katero

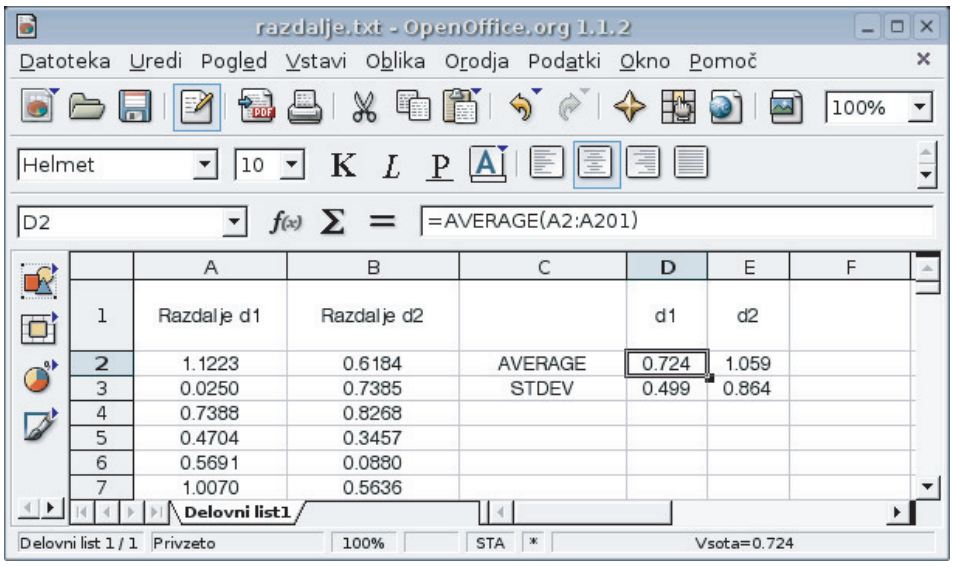
[2] Opozorilo: OpenOffice.org Calc je pisarniški program in ni namenjen znanstvenemu delu. Uporabljamo ga le zavoljo demonstracije interakcije med programskimi paketi.



Slika 11.3: Odpiranje datoteke `razdalje.txt`



Slika 11.4: Uvoz tekstovne datoteke v OpenOffice.org Calc



Slika 11.5: Izračun povprečja in standardne deviacije

zapišemo oznake stolpcev. V preglednici imamo sedaj dva stolpca: v stoplcu `A` so razdalje točk do prvega superkvadrika, v stolpcu `B` pa razdalje točk do drugega superkvadrika (slika 11.5).

Sedaj bomo modela statistično primerjali in izbrali boljšega od obeh. Najprej izračunamo povprečno razdaljo in standarden odklon za oba superkvadrika: v celico `D2` zapišemo formulo `=AVERAGE(A2:A201)`, v celico `D3` pa formulo `=STD(A2:A201)`. V prvi celici se prikaže vrednost povprečja vseh razdalj v prvem stolpcu, v drugi pa njihov standarden odklon. Obe formuli skopiramo v stolpec `E`, kot je bilo to opisano v poglavju o preglednicah, in tako dobimo rezultate še za drugi stolpec.

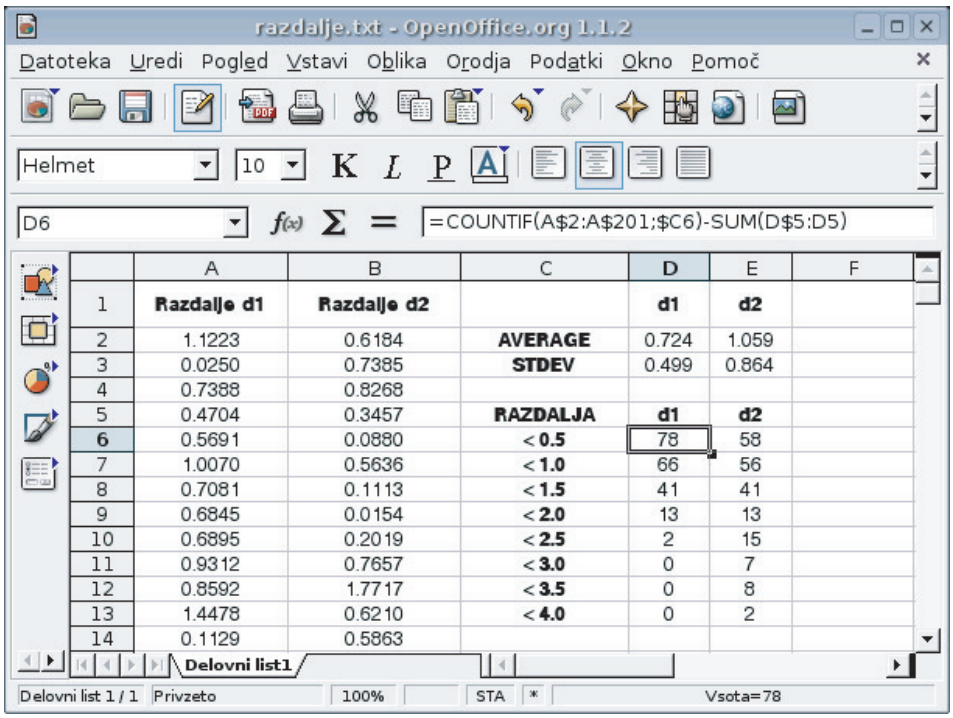
Takoj lahko vidimo, da prvi superkvadrik precej bolje opisuje oblak točk kot pa drugi, kateremu pripada večja povprečna razdalja in tudi večji standarden odklon. Za boljšo ponazoritev kakovostne razlike med modeloma bomo izdelali histogram. Kljub obilici statističnih funkcij, ki jih ponuja OpenOffice.org Calc, pa izdelava histograma nad podatki ni tako premočrtna naloga, kot bi si lahko kdo mislil.

Najprej bo potrebno določiti intervale histograma. S funkcijama `max` in `min` ocenimo meje zaloge vrednosti podatkov in se odločimo, da se bo histogram pričel z vrednostmi, manjšimi ali enakimi 0.5, in končal z vrednostmi, večjimi ali enakimi številu 4. Vmesni intervali pa naj bodo široki 0.5.

Intervale določimo z vpisi v polja `C6` do `C8` (slika 11.6). V celico `D6` nato vpišemo funkcijo, ki bo preštela vse celice `A2:A201`, ki ustrezajo pogoju v celici `C6`, nato pa odštela število vseh celic, ki ustrezajo vsem prejšnjim pogojem:

```
=COUNTIF(A$2:A$201;$C6)-SUM(D$5:D5)
```

Formulo nato skopiramo v celice `D6:E13`. Vrednosti v stolpcih sedaj predstavljajo histogram razdalj točk za oba superkvadrika.[3]



	A	B	C	D	E	F
1	**Razdalje d1**	**Razdalje d2**		**d1**	**d2**	
2	1.1223	0.6184	**AVERAGE**	0.724	1.059	
3	0.0250	0.7385	**STDEV**	0.499	0.864	
4	0.7388	0.8268				
5	0.4704	0.3457	**RAZDALJA**	**d1**	**d2**	
6	0.5691	0.0880	**< 0.5**	78	58	
7	1.0070	0.5636	**< 1.0**	66	56	
8	0.7081	0.1113	**< 1.5**	41	41	
9	0.6845	0.0154	**< 2.0**	13	13	
10	0.6895	0.2019	**< 2.5**	2	15	
11	0.9312	0.7657	**< 3.0**	0	7	
12	0.8592	1.7717	**< 3.5**	0	8	
13	1.4478	0.6210	**< 4.0**	0	2	
14	0.1129	0.5863				

Slika 11.6: Izračun histograma s funkcijo `COUNTIF`

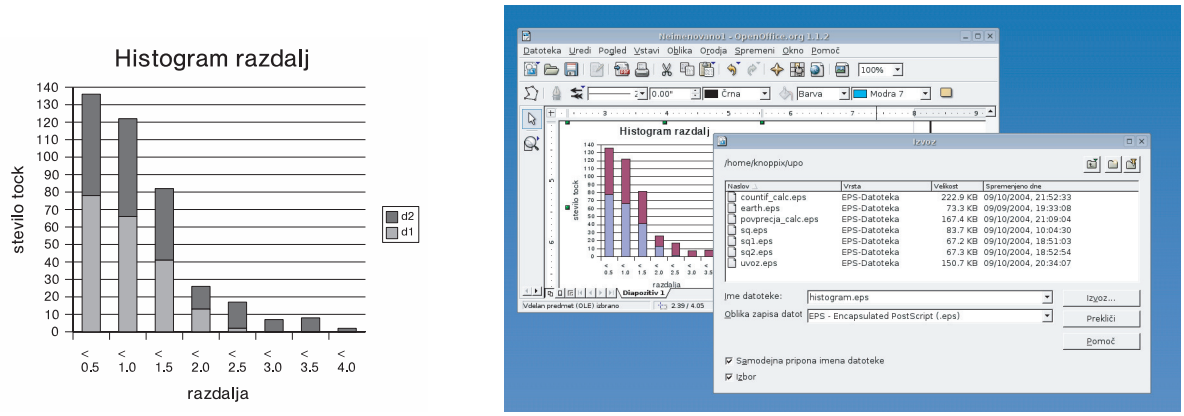
Grafiko histograma vstavimo v dokument tako, da označimo celice `C5:E13` in v meniju "Vstavi" izberemo opcijo "Grafikon". Na sliki 11.7(a) vidimo, da porazdelitev razdalj za prvi superkvadrik dejansko ponazarja večjo natančnost modela.

Histogram shranimo v vektorskem grafičnem formatu PostScript s pomočjo izbire "Datoteka→Izvoz" (slika 11.7(b)).

11.5 Izdelava poročila

Končno poročilo o našem delu bomo napisali s paketom za logično urejanje besedil LaTeX, ki je z leti postal že obvezen del vsake resnejše distribucije operacijskega sistema Linux. V poročilo bomo vstavili vektorske slike, ki smo jih med delom vestno shranjevali v zapisu PostScript. Dele programske kode bomo kopirali s postopkom izreži in

[3] Pogoji v celicah `C6:C13` ne opisujejo intervalov, temveč so le orodje za izračun histograma.



Slika 11.7: (a) Histogram in (b) izvoz histograma v format PostScript

prilepi (angl. *cut and paste*): iz datotek s končnico `.m` izrežemo potrebne dele kode in jih prilepimo v datoteko s končnico `.tex`, v kateri bomo ustvarili poročilo. Izvorna koda poročila je naslednja:

```
\documentclass[12pt,twocolumn,a4paper]{article}
\usepackage[slovene]{babel}
\usepackage{graphics,epsfig}

\begin{document}

\title{Superkvadriki}
\author{Peter Peer, Matja\v{z} Jogan}
\date{10. september, 2004}
\maketitle

\section{Definicija}

Superkvadriki~\cite{Kluwer} so geometrijska telesa, pri katerih je
vsaka to\v{c}ka na povr\v{s}ini definirana takole:

\begin{eqnarray}
          & x & = a_1\cos^{\epsilon_1}\eta \cos^{\epsilon_2} \omega,\\
\nonumber & y & = a_2{\cos^{\epsilon_1}\eta} \sin^{\epsilon_2}\omega,\\
\nonumber & z & = a_3\sin^{\epsilon_1}\eta.
\label{eq:b_matlab2_sq}
\end{eqnarray}

Spremenljivka $\eta$ lahko zavzame vrednosti med $-\pi/2$ in
$\pi/2$, $\omega$ pa med $-\pi$ in $\pi$. Parametri $a_1$, $a_2$
in $a_3$ dolo\v{c}ajo velikost superkvadrika, $\epsilon_1$ in
$\epsilon_2$ pa obliko.
```

```
\section{Radialna razdalja}

Radialno razdaljo to\v{c}ke do superkvadrika uporabljamo kot
pribli\v{z}ek evklidske razdalje. Radialna razdalja je definirana
kot:

$$
d=|\mathbf{r_0}||1-F^{-\frac{\epsilon_1}{2}}(x_0,y_0,z_0)|\mbox{ },
$$

\noindent kjer je

$$
F_{xyz}=\left(\left(\frac{x}{a_1}\right)^\frac{2}{\epsilon_2}+
\left(\frac{y}{a_2}\right)^\frac{2}{\epsilon_2}\right)^
\frac{\epsilon_2}{\epsilon_1}+
\left(\frac{z}{a_3}\right)^\frac{2}{\epsilon_1}\mbox{ }.
$$

\section{Ocena superkvadrika}

\begin{figure}[h]
\centerline{\psfig{figure=tocke.eps,width=5cm}}
\caption{Oblak to\v{c}k}
\label{tocke}
\end{figure}

Podana imamo dva superkvadrika, ki opisujeta oblak to\v{c}k na
sliki~\ref{tocke}. Parametri prvega so $a^1_1=15$, $a^1_2=15$,
$a^1_3=20$, $\epsilon^1_1=1.8$ in $\epsilon^1_2=1.5$
(slika~\ref{superkvadrika} (a)), parametri drugega pa $a^2_1=12$,
$a^2_2=12$, $a^2_3=20$, $\epsilon^2_1=1.5$ in $\epsilon^2_2=1.2$
(slika~\ref{superkvadrika} (b)). Oceniti moramo, kateri
superkvadrik bolje opisuje mno\v{z}ico 3D to\v{c}k.

\begin{figure}[h]
\centerline{
\psfig{figure=superkvadrik1.eps,width=4cm}
\psfig{figure=superkvadrik2.eps,width=4cm}}
\centerline{\hfill (a)\hspace{3.5cm}(b)\hfill}
\caption{Superkvadrika in oblak to\v{c}k v istem koordinatnem sistemu}
\label{superkvadrika}
\end{figure}


Oceno osnujemo na statistiki razdalj oblaka to\v{c}k do
povr\v{s}ine superkvadrika. Razdalje izra\v{c}unamo tako, da izraz
za radialno razdaljo kodiramo v programskem jeziku Octave.

Odstopanje obeh superkvadrikov od danih 3D to\v{c}k grafi\v{c}no
ponazorimo s histogramom razdalj, ki ga pripravimo in prika\v{z}emo
```

```
v programu OpenOffice.org Calc (slika \ref{histogram}).

V programu OpenOffice.org Calc iz\-ra\-\v{c}u\-namo tudi
povpre\v{c}no razdaljo in standarden odklon razdalj za oba
superkvadrika. Rezultate vidimo v tabeli~\ref{tabela1}.

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|}
\hline  & d1  & d2 \\ \hline \hline
 povpre\v{c}je & 0.724 & 1.059 \\
\hline
 stand. odkl. & 0.499 & 0.864  \\
\hline
\end{tabular}
\caption{Povpre\v{c}je in standarden odklon razdalj to\v{c}k do
superkvadrikov} \label{tabela1}
\end{table}

Tako histogram kot tudi analiza statisti\v{c}nih momentov
ka\v{z}ejo na to, da prvi superkvadrik precej bolje opisuje oblak
to\v{c}k kot pa drugi, kateremu pripada ve\v{c}ja povpre\v{c}na
razdalja in tudi ve\v{c}ji standarden odklon.

\begin{figure}[tbh]
\centerline{\psfig{figure=histogram.eps,width=8cm}}
\caption{Histogram razdalj}
\label{histogram}
\end{figure}

\section{Uporabljena orodja}

Pri delu sva uporabila programa Octave in OperOffice.org Calc,
to poro\v{c}ilo je izdelano s sistemom \LaTeX, predstavitev
dela pa je izdelana v programu OpenOffice.org Impress \cite{UPO}.

\begin{thebibliography}{1}
\bibitem{Kluwer} A. Jakli\v{c}, A. Leonardis, and F. Solina,
\textit{Segmentation and Recovery of Superquadrics}, Kluwer,
Dordrecht, 2000.
\bibitem{UPO}
M. Jogan, A. Leonardis, I. Lesjak, B. Kverh, P. Peer, F. Solina,
\textit{Uporabni\v{s}ka programska oprema}, Fakulteta za
ra\v{c}unalni\v{s}tvo in informatiko, Ljubljana, 2001.
\end{thebibliography}

\end{document}
```

Superkvadriki

Peter Peer, Matjaž Jogan

10. september, 2004

1 Definicija

Superkvadriki [1] so geometrijska telesa, pri katerih je vsaka točka na površini definirana takole:

$$\begin{aligned} x &= a_1 \cos^{\epsilon_1} \eta \cos^{\epsilon_2} \omega, \\ y &= a_2 \cos^{\epsilon_1} \eta \sin^{\epsilon_2} \omega, \\ z &= a_3 \sin^{\epsilon_1} \eta. \end{aligned} \tag{1}$$

Spremenljivka η lahko zavzame vrednosti med $-\pi/2$ in $\pi/2$, ω pa med $-\pi$ in π. Parametri a_1, a_2 in a_3 določajo velikost superkvadrika, ϵ_1 in ϵ_2 pa obliko.

2 Radialna razdalja

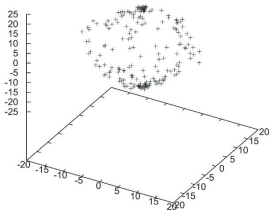
Radialno razdaljo točke do superkvadrika uporabljamo kot približek evklidske razdalje. Radialna razdalja je definirana kot:

$$d = |\mathbf{r_0}||1 - F^{-\frac{\epsilon_1}{2}}(x_0, y_0, z_0)| \ ,$$

kjer je

$$F_{xyz} = \left(\left(\frac{x}{a_1} \right)^{\frac{2}{\epsilon_2}} + \left(\frac{y}{a_2} \right)^{\frac{2}{\epsilon_2}} \right)^{\frac{\epsilon_2}{\epsilon_1}} + \left(\frac{z}{a_3} \right)^{\frac{2}{\epsilon_1}} .$$

3 Ocena superkvadrika



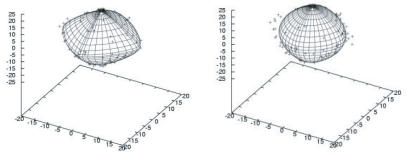
Slika 1: Oblak točk

Podana imamo dva superkvadrika, ki opisujeta oblak točk na sliki 1. Parametri prvega so $a_1^1 = 15$, $a_2^1 = 15$, $a_3^1 = 20$, $\epsilon_1^1 = 1.8$ in $\epsilon_2^1 = 1.5$ (slika 2 (a)), parametri drugega pa $a_1^2 = 12$, $a_2^2 = 12$, $a_3^2 = 20$, $\epsilon_1^2 = 1.5$ in $\epsilon_2^2 = 1.2$ (slika 2 (b)). Oceniti moramo, kateri superkvadrik bolje opisuje množico 3D točk.

Oceno osnujemo na statistiki razdalj oblaka točk do površine superkvadrika. Razdalje izračunamo tako, da izraz za radialno razdaljo kodiramo v programskem jeziku Octave.

Odstopanje obeh superkvadrikov od danih 3D točk grafično ponazorimo s histogramom razdalj, ki ga pripravimo in prikažemo v programu OpenOffice.org Calc (slika 3).

Slika 11.8: Prva stran poročila



(a) (b)

Slika 2: Superkvadrika in oblak točk v istem koordinatnem sistemu

V programu OpenOffice.org Calc izračunamo tudi povprečno razdaljo in standarden odklon razdalj za oba superkvadrika. Rezultate vidimo v tabeli 1.

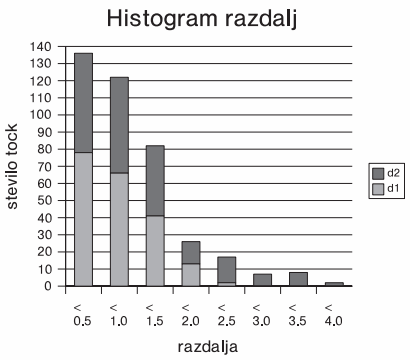
	d1	d2
povprečje	0.724	1.059
stand. odkl.	0.499	0.864

Tabela 1: Povprečje in standarden odklon razdalj točk do superkvadrikov

Tako histogram kot tudi analiza statističnih momentov kažejo na to, da prvi superkvadrik precej bolje opisuje oblak točk kot pa drugi, kateremu pripada večja povprečna razdalja in tudi večji standarden odklon.

4 Uporabljena orodja

Pri delu sva uporabila programa Octave in OperOffice.org Calc, to poročilo je izdelano s sistemom LaTeX, predstavitev dela pa je izdelana v programu OpenOffice.org Impress [2].



Slika 3: Histogram razdalj

Literatura

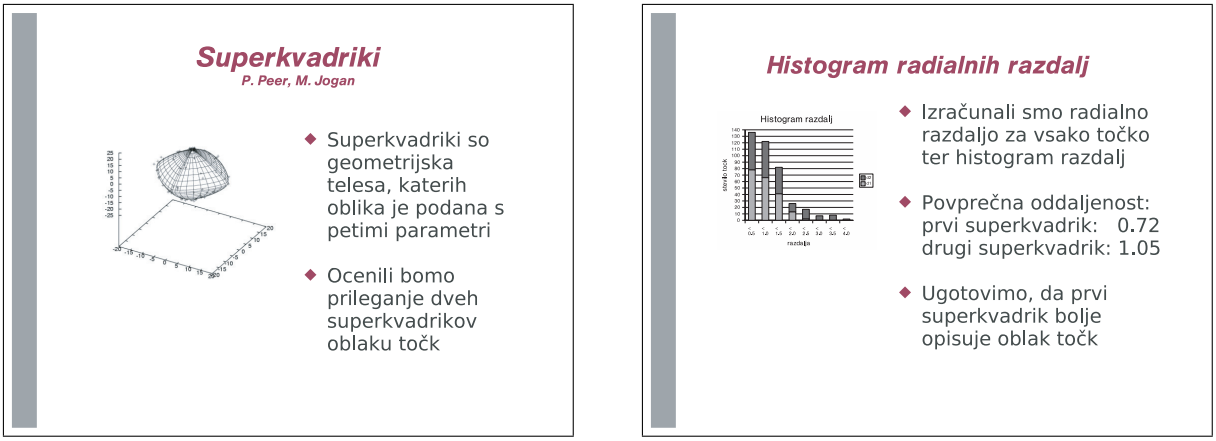
[1] A. Jaklič, A. Leonardis, and F. Solina, *Segmentation and Recovery of Superquadrics*, Kluwer, Dordrecht, 2000.

[2] M. Jogan, A. Leonardis, I. Lesjak, B. Kverh, P. Peer, F. Solina, *Uporabniška programska oprema*, Fakulteta za računalništvo in informatiko, Ljubljana, 2001.

Slika 11.9: Druga stran poročila

11.6 Izdelava predstavitve

Potrebujemo še predstavitev, ki jo izdelamo v programu OpenOffice.org Impress. Potrebne slike PostScript uvozimo neposredno, enačbe pa zapišemo v urejevalniku enačb, dosegljivim preko izbire "Vstavi→Objekt→Formule". Dve strani predstavitve vidimo na sliki 11.10.



Slika 11.10: Predstavitev v OpenOffice.org Impress

11.7 Literatura

[1] J. N. Bronštejn and K. A. Semendjajev. *Matematični priročnik*. Tehniška založba Slovenije, Ljubljana, 1988.

[2] A. Jaklič, A. Leonardis, and F. Solina. *Segmentation and Recovery of Superquadrics*. Kluwer, Dordrecht, 2000.

[3] D. Skočaj. Avtomatsko modeliranje 3-dimenzionalnih predmetov z uporabo globinskega senzorja. Magistrsko delo iz računalništva in informatike, Fakulteta za računalništvo in informatiko, Univerza v Ljubljani, Ljubljana, 1999.

Dodatek A

Delo s Slixom in namestitev Linuxa

Priložena zgoščenka vsebuje popolno distribucijo Linuxa pod imenom Slix, ki jo razvija multimedijski kulturni center Ljudmila. Kot pravijo avtorji distribucije (`http://slix.ljudmila.org/sl/about/`), je Slix distribucija Linuxa za nekomplicirane uporabnike računalnikov, ki se izvaja kar z zgoščenke, kar pomeni, da namestitev na trdi disk računalnika sploh ni potrebna. Namestitev na trdi disk je seveda možna, vendar pa velja, ker distribucija temu ni namenjena, za resnejše in dolgoročno delo razmisliti o kaki namenski distribuciji. Slix je uporaben tudi kot univerzalen terminal za dostop do omrežnih storitev ali kot zgoščenka za diagnostiko in morebitna popravila.

A.1 Slix – kako naj shranim podatke?

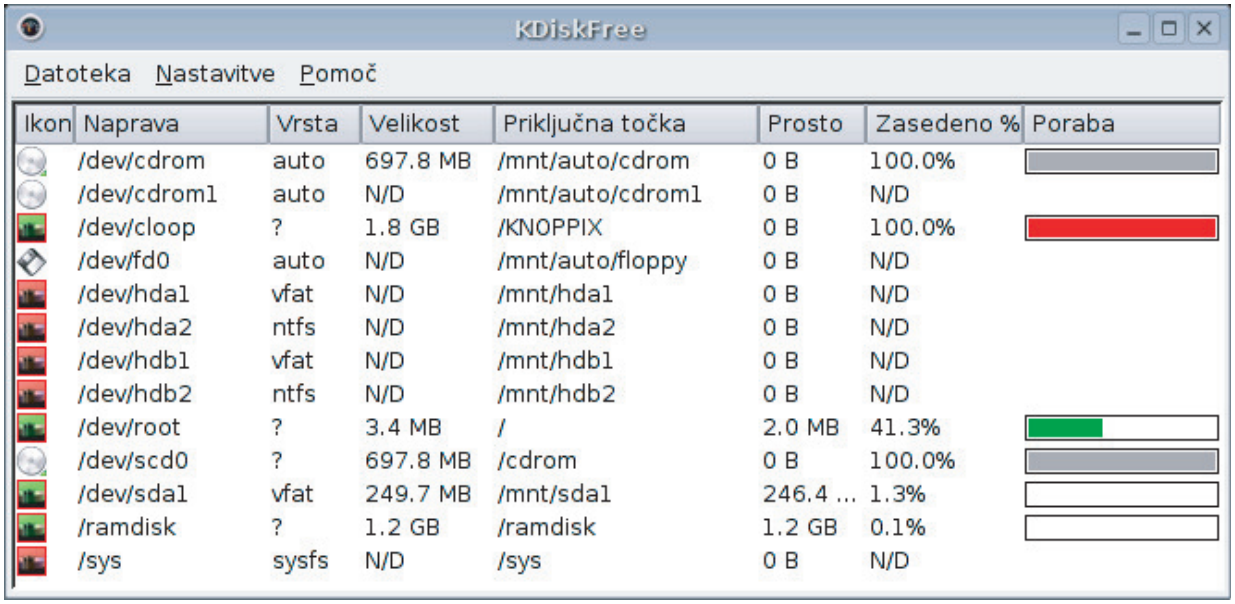
Uporaba Slixa je tudi za neizkušenega uporabnika enostavna. Dovolj je, da v CD pogon vstavimo zgoščenko z Slixom in (ponovno) poženemo računalnik. Operacijski sistem bo med nalaganjem zaznal večino naprav in jih samodejno nastavil. Pred delom bo morda potrebno nastaviti le jezik (Slix in kar nekaj programskih paketov je prevedenih tudi v slovenščino), tipkovnico in vrsto namizja.

Slix ob zagonu ustvari virtualni podatkovni prostor v spominu, kamor shranjuje uporabnikove datoteke. Ker Slix samodejno razpozna naprave, zazna tudi diskovje. Če ima naš računalnik particijo, na katero zna Slix pisati in brati (ext2, ext3 ali vfat), jo lahko enostavno namestimo (glej poglavje o operacijskih sistemih) in uporabljamo. Vendar pa računalniki, ki imajo nameščen oprecijski sistem Windows XP ali Windows 2000, ponavadi take particije nimajo.

Če želimo po končanem delu vseeno ohraniti svoje podatke (brez

prenašanja na oddaljeni računalnik, denimo preko protokola FTP), si lahko pomagamo z uporabo izmenjivih pomnilnikov. Enostaven izmenljiv pomnilnik je disketa, a je zaradi zastarelosti, počasnosti in majhne zmogljivosti slaba izbira.

Priporočljiva izbira za povprečnega uporabnika je pomnilniški ključ USB, ki predstavlja v času pisanja te knjige klasično izbiro za hitro in varno shranjevanje podatkov. Slix samodejno prepozna pomnilniške ključe USB 1.0 in USB 2.0, tako da jih je potrebno pred uporabo le aktivirati. Pri tem si lahko pomagamo s programom KwikDisk, ki je kot naročen za upravljanje z izmenljivimi diski. V njem lahko s klikom na miško enostavno priklapljamo in odklapljamo naprave (slika A.1).



Slika A.1: Upravljanje z izmenljivimi zunanjimi pomnilniki s programom KwikDisk

Preko terminala lahko svojo delovno mapo nastavimo takole: ob zagonu Slixa je naša domača mapa `/home/knoppix`. V tej mapi lahko naredimo novo delovno mapo (npr. `UPO`), ki bo predstavljala povezavo do naprave USB, ki jo Linux označuje kot `/mnt/sda1`:

```
>> mount /mnt/sda1
>> ln -s /mnt/sda1 UPO
```

Datoteke v tej mapi se bodo fizično nahajale na USB pomnilniškem ključu in bodo tam ostale tudi po izklopu računalnika. Sedaj lahko mapo UPO uporabljamo kot delovno mapo. Vendar pa se osebne nastavitve, shranjene v mapi `/home/knoppix`, ne bodo ohranile.

Sistem Knoppix, na katerem je Slix zasnovan, ponuja rešitev, ki omogoča, da na izmenljivem pogonu ustvarimo dejansko domačo mapo, ki hrani tudi vse osebne sistemske nastavitve in deluje transparentno kot običajna domača mapa:

ob zagonu računalnika vstavimo ključ USB, vendar ga ne aktiviramo. V glavnem menuju izberemo opcijo "K→KNOPPIX→Configure→Create a persistent KNOPPIX Home directory". Nato izberemo `/mnt/sda1` (ali kako drugo izmenljivo enoto) in določimo, koliko prostora bomo namenili za domačo mapo (tu je ponujenih 30Mb dovolj, saj lahko ostanek prostora na ključu še vedno uporabljamo za shranjevanje datotek). Domača mapa bo na ključu spravljena v kodirani datoteki `knoppix.img`.

Pri zagonu Slixa moramo sedaj podati lokacijo domače mape:

```
boot: knoppix home=/mnt/sda1
```

ali pa vnesemo

```
boot: knoppix home=scan
```

kar bo sprožilo samodejno iskanje domače mape. Zanimiv vidik dela s takšnim sistemom je ta, da potrebujemo le zgoščenko in dovolj zmogljiv ključ USB in že lahko vsak PC z vodilom USB spremenimo v svoj domač računalnik.

A.2 Namestitev Linuxa

Namestitev operacijskega sistema je precej neprijetno opravilo. Ta je bila pri starejših distribucijah Linuxa še posebej zapletena, v novih distribucijah pa je precej poudarka ravno na prijaznosti do uporabnika, zato je osnovna namestitev na prazno diskovje praviloma neproblematična. Problemi pa se pojavijo, če želimo imeti hkrati nameščena dva sistema, Windows in Linux. Zato bomo na kratko opisali splošna pravila namestitve. Ker se podrobnosti med različnimi distribucijami razlikujejo, bomo opisali le najbolj osnovne postopke. Natančnejša navodila za namestitev najdemo v [1].

Na računalniku je že nameščen OS Windows

Operacijski sistem Linux lahko namestimo tudi na računalnik, ki že ima nameščen drug operacijski sistem (na primer Windows). Običajno se ta nahaja na particiji, ki zaseda ves prostor na disku. V tem primeru si lahko pomagamo na dva načina.

Prva možnost je ponovno particioniranje diska. V tem primeru je potrebno izvesti dva postopka.

- Najprej moramo podatke, ki so razpršeni po celotnem disku, s posebno programsko opremo prestaviti na začetek particije (zgostiti).
- Nato lahko disk ponovno particioniramo, tako da obsoječo particijo skrčimo in pustimo nekaj praznega prostora za Linux.

Za oba opisana postopka obstaja veliko različnih orodij.[1] Velja opozoriti, da je particioniranje nepovraten postopek. Če so podatki na particiji brezhibni, je strah pred izgubo podatkov načeloma odveč. Izgubijo se običajo le že izbrisane datoteke (ki jih je pred particioniranjem še moč povrniti). Če pa so na particiji napake v njenem datotečnem sistemu, na primer nepovezane verige gruč (angl. *cluster*), nastopijo težave že pri zgoščevanju. Zato je pred zgoščevanjem in ponovnim particioniranjem priporočljiva izdelava rezervne kopije podatkov na obstoječih particijah.

Rezultat ponovnega particioniranja je disk, ki ima na primer eno primarno particijo z datotečnim sistemom FAT32 ali NTFS (uporablja jo Windows) in del diska, ki je ostal nerazporejen.

Če particija FAT zaseda celotni trdi disk, lahko uporabimo tudi namestitev brez particioniranja. V tem primeru ni potrebno ustvarjanje particij z Linuxovim datotečnim sistemom, saj se ta namesti na obstoječo particijo FAT. Namestitev brez particioniranja pa ima nekatere omejitve:

- Namestitev programa za nalaganje operacijskega sistema LILO ni možna. Za zagon Linuxa je potrebna zagonska disketa, ki se ustvari na koncu namestitve.
- Učinkovitost sistema je slabša kot v primeru, če naredimo prostor za Linuxove particije.

Ne glede na to, ali uporabimo Linuxove particije, moramo izvršiti popoln postopek namestitve, ki se (razen v delu, namenjenemu particioniranju) ne razlikuje od običajnega postopka namestitve.

Na disku še ni operacijskega sestema

V primeru, da na disku še ni nameščenega operacijskega sistema, je potrebno disk najprej pripraviti za oba operacijska sistema. Pripravimo zagonsko disketo, na katero dodamo pogram, ki omogoča spreminjanje

[1] Veliko orodij je zbranih v zbirki, ki je na voljo na naslovu `http://www.ultimatebootcd.com/`.

particijske tabele.[2] Primarno particijo z datotečnim sistemom FAT32 ali NTFS pripravimo za Windows, ostali del diska pa pustimo nerazporejen. Praviloma operacijski sistem Windows namestimo pred Linuxom.

Namestitev

Ko smo na disku naredili prostor za namestitev Linuxa, lahko pričnemo z namestitvijo. Pri običajnem postopku namestitve se bo Linux namestil na nerazporejeni del diska in ga razkosal na več particij s svojim datotečnim sistemom. Iz Linuxa lahko dostopamo tudi do podatkov na particiji z datotečnima sistemoma FAT ali FAT32, Windows pa Linuxovega datotečnega sistema EXT2 ne podpira. Linux bo ob namestitvi preveril glavni začetni zapis in zaznal že nameščeni operacijski sitem. Po privzetih nastavitvah se v glavni začetni zapis namesti program za izbiro operacijskega sistema *LILO boot*, kar uporabniku ob zagonu omogoča izbor enega od obeh operacijskih sistemov.

Prostor, ki je potreben za namestitev operacijskega sistema Linux, je odvisen od vrste namestitve, ki jo izberemo. Paketi, ki jih običajni uporabnik najpogosteje potrebuje za svoje delo, zasedejo okoli 850 MB prostora na disku. Če ne izberemo ročne nastavitve particij, bo sistem ponavadi samodejno opravil particioniranje in namestil LILO. V primeru, da želimo po zaključenem postopku namestitve dodati nove ali izbrisati katerega od nameščenih paketov, moramo to narediti sami z enim od priloženih programov (na primer s `kpackage`).

Dokler različica Linuxa, ki jo nameščamo, podpira vso strojno opremo, ki je nameščena v računalniku, je postopek namestitve precej preprost. Težave lahko nastanejo s strojno opremo, ki je novejša od različice operacijskega sistema, ali pa z opremo, ki je Linux ne podpira. Če naprave, ki je Linux ne pozna, ne potrebujemo pri delu v Linuxu, nas to ne moti. Ko pa napravo potrebujemo, je pomoč potrebno poiskati na svetovnem spletu, kjer v večini primerov najdemo gonilnike in navodila za namestitev.

Čeprav bo Linux v postopku namestitve pravilno zaznal večino strojne opreme, se lahko na njegovo namestitev pripravimo tako, da zberemo podatke o strojni opremi, nameščeni v računalniku. Če imamo nameščen OS Windows, si lahko pomagamo s podatki, ki jih je zbral o strojni opremi.

[2] Še bolje je, če iz zgornjega spletnega mesta naložimo in spečemo zagonsko zgoščenko.

A.3 Namestitev Slixa na disk

Slix je v osnovi sistem, namenjen poganjanju z zgoščenke. Posebnost Slixa je prav možnost, da do popolnega operacijskega sistema dostopamo brez neposredne namestitve na disk. Knoppix, na katerem je Slix zasnovan, skrbi za sprotno kompresijo datotek in upravljanje z virtualnim pomnilnikom in diskom. Če pa vseeno želimo Slix namestiti na diskovje, postane podoben distribuciji Debian.

Za instalacijo Slixa na diskovje potrebujemo program `knoppix-inst`. Ker ga nekatere verzije Slixa nimajo na voljo, ga lahko pretočimo z naslova razvijalcev. V terminalsko ukazno vrstico vtipkamo:

```
> FILE=$(wget http://debian.tu-bs.de/knoppix/installer/latest -O -)
> echo "Downloading $FILE ..." cd $HOME wget
> http://debian.tu-bs.de/knoppix/installer/$FILE tar xzf $FILE cd
> knx-installer-* sudo ./knoppix-installer
```

Na voljo imamo več namestitev:

1. namestitev Knoppix na trdi disk presname zgolj kopijo zagonske zgoščenke in omogoča udobnejšo uporabo Slixa, a z vsemi pomanjkljivostmi CD distribucije – brez opcij večuporabniškega sistema in neprilagojeno omežnemu okolju;
2. namestitev Debian, ki je najbolj podobna standardni Debian namestitvi, a z nekaj pomanjkljivostmi;
3. namestitev Beginner default, namenjena začetnikom, ima lastnosti pravega Linuxa (npr. večuporabništvo), vendar pa uporablja Knoppixovo samodejno detekcijo naprav.

Namestitveni program lahko poženemo tudi tako, da s kombinacijo tipk `<CTRL>+<ALT>+<F1>` vstopimo v nadzorno konzolo in tam vtipkamo:

```
> knx-hdistall
```

Pogovorna okna nas bodo vodila skozi naslednje stopnje namestitve:

- ustvarjanje Linuxove particije na disku
- ustvarjanje izmenjalne particije `swap` (najmanjša velikost 256 Mb)
- nastavljanje Linuxove `root` particije
- kopiranje potrebnih datotek

- nastavljanje mrežnih nastavitev
- nastavitev gesel
- nastavitev programa za nalaganje operacijskega sistema
- ponovni zagon računalnika

O Linuxovih particijah je potrebno vedeti naslednje: nujno moramo ustvariti najmanj dve particiji: `root` in `swap`. Slednja sicer ni obvezna, saj lahko njeno nalogo opravlja pomnilnik, a je priporočljiva. Particija `root` naj se ne bi nahajala pod prvimi 1023 cilindri. Razlog za to je, da se nekateri BIOS-i ne omogočajo zagona sistema, nameščenega pod prvimi 1023 cilindri. Veljalo je tudi, naj bo velikost particije `swap` približno dvakrat večja, kot je količina spomina v spominskih čipih, a to je držalo le za sisteme z manj kot 100Mb spomina.

Zavoljo robustnosti sistema navadno ločimo particijo `root` od ostalih – namenimo ji le toliko prostora, kot je nujno potrebno. Ostale dele datotečnega sistema razporedimo po lastni zamisli – pogosto npr. dodelimo večji del diska izključno mapi `home`.

A.4 Koristne spletne povezave

1. Namestitev in začetek dela z Linuxom v slovenščini:
   `http://www.lugos.si/delo/slo/LIGS-sl/`
2. Domača stran projekta Slix:
   `http://slix.ljudmila.org/sl/`
3. Domača stran sistema Knoppix:
   `http://www.knoppix.net`
4. Domača stran Linuxa Red Hat (Fedora):
   `http://www.redhat.com/`

A.5 Literatura

[1] A. Košir, R. Maurer, P. Peterlin, and M. Samastur. *Namestimo Linux.* Pasadena, Ljubljana, 2001.

Dodatek B

Slovarček

account	uporabniško dovoljenje, račun
adware	programje podprto z oglaševanjem
ambient intelligence	okoljska inteligenca
bitmap graphics	rastrska računalniška grafika
bold font	krepka pisava
bookmark	zaznamba
browser	brskalnik
CD	zgoščenka
chapter	poglavje
clipboard	odložišče
cluster	gruča
commercial software	komercialno programje
context menu	kontekstni meni
counterfeiting	ponarejanje
cut and paste	izreži in prilepi
decorative scripts	dekorativne ali akcidenčne pisave
demo version	demonstracijska različica
desktop	namizje
dialog	pogovorno okno
directory	imenik, direktorij, mapa
drag and drop	povleci in spusti
drawing	risba
editor	urejevalnik
effect	učinek
field	polje
file	datoteka
file manager	upravljalec datotek
floating window	plavajoče okno

font	pisava
footer	vznožje strani
frame	okvir
free software licences	prosto programje
freeware	zastonjsko programje
function toolbar	funkcijska orodjarna
grid	rešetka
header	vrh strani, informacija na začetku datoteke
help	pomoč
HTML tag	oznaka HTML
image	slika
image depth	globina slike (običajno v bitih)
interface	vmesnik
interpreter	tolmač
italic font	kurziva
kernel	jedro
kerning	spiranje
layer	plast
link	povezava
mail client	poštni odjemalec
mail server	poštni strežnik
mailbox, inbox	poštni predal
mailer	odjemalec za elektronsko pošto
margin size	širina roba
markup	označevanje
monospaced font	neproporcionalna pisava
morphing	prehajanje
mount	priklopiti, namestiti
mounted	priklopljen, nameščen
multimedia	večpredstavnost
multitasking	večopravilni način delovanja
object toolbar	predmetna orodjarna
open source	odprto programje
OS shell	lupina operacijskega sistema
outline	osnutek
outline fonts	obrisne črke
paragraph	odstavek
parser	razpoznavalnik
password	geslo
pixel	slikovni element

preformated	vnaprej oblikovano
progress bar	prikaz poteka
proprietary software	lastniško programje
public domain software	programje v javni lasti
pulldown menu	izvlečni meni
range image	globinska slika
record	zapis
resolution	ločljivost
roman font	pokončna pisava
root	korenski imenik
sans serif font	linearna pisava
scroll bar	drsni trak
section	razdelek
slider	drsnik
spam	nezaželena reklamna elektronska pošta
spreadsheet	preglednica
square serif types	pisave z oglatimi serifi
small caps	male kapitelke
socket	vtičnica
softlifting	mehko piratstvo
stack	sklad
status bar	statusna vrstica, vrstica stanja
style	slog
subtitle	podnaslov
tab	zahivek
tag	označba, oznaka
task	opravilo
task bar	opravilna vrstica
template	vzorec, predloga
toolbox, toolbar	orodjarna
trial version	poskusna različica
unmount	odklopiti
user name	uporabniško ime
version	različica
virtual desktop	delovno področje, navidezno namizje
virtual reality	virtualna realnost
web browser	spletni brskalnik
web page	spletna stran
web presentation	spletna predstavitev
web search	iskanje po spletu

web site	spletno mesto
window manager	upravljalec oken
worksheet	delovni list